UNIVERSIDAD DE BUENOS AIRES

Facultad de Ciencias Exactas y Naturales

Departamento de Matemática

Tesis de Licenciatura

Métodos robustos para clustering de series de tiempo

Juan Sebastián Castaño Cruz

Director/a: Dr. Martín Maas

2024-08-17

Índice General

Abstract en Castellano

En esta tesis se investigan métodos robustos de clustering y técnicas de preprocesamiento de series de tiempo con valores atípicos y/o datos faltantes. Además, se analizan técnicas de extracción de características (feature extraction) de series temporales.

La motivación del desarrollo de esta metodología es la identificación del área sembrada con distintos cultivos en Argentina, utilizando series de imágenes satelitales que proporcionan series de tiempo del índice de vegetación NDVI en el territorio agrícola. La metodología propuesta incluye el tratamiento de valores atípicos en múltiples series de tiempo del índice NDVI, la extracción de características y el uso de aprendizaje no supervisado robusto para la detección de patrones en las series temporales. Se estudian, en particular, el filtro de Hampel basado en el estimador MAD, y el algoritmo TCLUST como método de clustering robusto. El trabajo presenta los diferentes aspectos metodológicos junto con ejemplos con datos simulados.

Finalmente, se aplica la metodología para realizar una predicción de la proporción de trigo cultivado en Argentina a escala departamental, para el año 2021. En el análisis realizado sobre un total de 220 departamentos, el error cuadrático ponderado por superficie obtenido con clustering no robusto para estimar las proporciones, comparado con las proporciones reales, fue de 7.2% rms . En contraste, al aplicar clustering robusto, este error se redujo a 5.9% rms, evidenciando una mejora sustancial en la precisión de las estimaciones. Como trabajo futuro, se propone extender esta metodología para predecir de manera más eficiente y viable las proporciones de otros cultivos de relevancia económica y alimentaria en el país, como la soja y el maíz.

Abstract in English

This thesis investigates robust clustering methods and techniques for preprocessing time series with outliers and/or missing data. Additionally, it analyzes feature extraction techniques for time series.

The motivation behind developing this methodology is to identify the area planted with different crops in Argentina, using time series of the NDVI vegetation index obtained from satellite images of the agricultural territory. The proposed methodology includes outlier treatment in multiple NDVI time series, feature extraction, and the use of robust unsupervised learning for pattern detection in time series. In particular, the Hampel filter based on the MAD estimator and the TCLUST algorithm as a robust clustering method are studied. The work presents various methodological aspects along with examples using simulated data.

Finally, the methodology is applied to predict the proportion of wheat cultivated in Argentina at the departmental level for the year 2021. In the analysis conducted across a total of 220 departments, the surface-weighted mean square error obtained using non-robust clustering to estimate the proportions, compared to the actual proportions, was 7.2% rms. In contrast, applying robust clustering reduced this error to 5.9% rms, demonstrating a substantial improvement in the accuracy of the estimates. As future work, it is proposed to extend this methodology to predict more efficiently and viably the proportions of other crops of economic and food relevance in the country, such as soybeans and corn.

1 Introducción

En la era digital actual, estamos inmersos en un mar de datos generados constantemente por una variedad de fuentes, desde dispositivos móviles y sensores hasta redes sociales y sistemas empresariales. Este crecimiento exponencial de datos ha creado una necesidad urgente de herramientas y técnicas que nos permitan extraer conocimientos significativos y tomar decisiones informadas a partir de estos datos. En este contexto, las técnicas estadísticas clásicas, pueden (y deben) ser combinadas con el poder de cómputo disponible en la actualidad para lograr aprovechar la información contenida en grandes volúmenes de datos.

Una de las fuentes de datos de mayor volúmen son los datos producidos por los sitemas orbitales de observación terrestre. Las imágenes satelitáles obtenidas permiten extraer información sobre características químicas, físicas y biológicas del planeta, algunas de ellas difícilmente visibles desde el suelo o a baja altura. Sin embargo, los datos efectivamente medidos (reflectancias en distintas longitudes de onda) no necesariamente se corresponden de manera directa con las variables de interés (como el impacto de la sequía en una zona agrícola determinada). Para poder convertir la información medida en variables de interés para un usuario final, es necesario construir un complejo *sistema de inferencia*, que incluye desde modelos físicos, pasando por métodos estadísticos clásicos, y, durante los últimos años, se han ido sumando numerosas técnicas de estadística computacional o *machine learning*.

Una de las características que presentan los datos observados es la presencia de datos atípicos. Dichos fenómenos pueden volver inaplicables a muchas técnicas. Por ejemplo, en el caso de los datos satelitales, al observar una serie de tiempo de reflectancias de un cultivo, la misma serie puede verse contaminada por días con presencia de nubes que no han logrado ser corregidas con procedimientos específicos, u otros fenómenos que pueden causar errores de medición. Para lidiar con estas características de los datos, existe la llamada estadística robusta, cuyo objetivo es producir estimadores que no sean afectados por una cantidad de datos que presenta grandes variaciones respecto a las hipótesis de los modelos.

Otro problema recurrente, que puede sonar paradójico en la era actual de gran abundancia de datos, es el problema de los datos faltantes. En muchas aplicaciones, como en el estudio de series de tiempo, una de las hipótesis básicas suele ser la disponibilidad de datos para un intervalo de tiempo fijo dado. En un marco donde no se satisfaga esta hipótesis se requirerá *imputar* ese dato faltante o recurrir a redefinir el problema en uno distinto.

Finalmente, podemos mencionar el problema de la extracción de parámetros. Muchos modelos, por ejemplo los modelos de clasificación, requieren que estén definidos una serie de parámetros los cuales extraen la naturaleza de los datos y pueden también reducir la dimensionalidad de los mismos.

En esta tesis abordaremos algunos métodos robustos para el problema de la clasificación de series de tiempo. Se considerarán todos los aspectos necesarios para su aplicación a datos

reales: desde el técnicas de pre-procesamiento para identificar y eliminar datos atípicos, técnicas para extraer parámetros en presencia de datos faltantes, y, finalmente, métodos robustos para el problema de clasificación no-supervisada o clústering.

Finalmente, presentaremos una aplicación a la detección de cultivos sembrados con trigo en Argentina, mediante un conjunto de datos satelitales de alta resolución. Los datos considerados, fueron medidos por el sistema Sentinel-2, y tienen una resolución de $10m^2$. El área considerada es, a su vez, un total de 220 departamentos en 9 provincias del país, alcanzando un poco más de $1200000Km^2$. Esto nos da un dataset del orden de los $40GB$, cuya manipulación nos obliga a tener en cuenta ciertos aspectos de la eficiencia computacional de los métodos escogidos.

1.1 Especificación del Problema

La utilización de datos satelitales en Latinoamérica reviste una gran importancia, dado que muchos países, entre ellos la Argentina, poseen un extenso territorio muy poco monitoreado por medios tradicionales, como estaciones meteorológicas u otros tipos de sensores en el terreno. En el caso de Argentina, uno de los problemas más importantes a abordar mediante información satelital a gran escala es la alerta temprana de sequías (Salvia et al. 2021).

Por medio de imágenes satelitáles y del aprendizaje automático predeciremos y agruparemos tendencias en los cultivos para los años desde el 2017 hasta el 2022 en el territorio agrícola Argentino, el cual contiene vastas, fértiles y escasamente monitoreadas tierras de cultivo, responsables de una parte importante de la producción alimentaria mundial (Rodríguez et al. 2024). El análisis de datos satelitales junto con distintas técnicas de aprendizaje de Máquina son necesarios para la eficiencia en los análisis de grandes áreas de terreno, en las que se involucren una gran cantidad de datos. Se emplearán métodos de Aprendizaje No supervisado para el análisis y estudio de los cultivos.

Para la especificación del problema que queremos abordar queremos hacer clustering para un enorme conjunto de datos cuyas observaciones corresponden a series de tiempo del Índice de Vegetación de Diferencia Normalizada (NDVI por sus siglas en inglés) de la región considerada. Se busca encontrar patrones entre agrupaciones de series de tiempo que nos permitan entender los distintos tipos de cultivos y en particular la proporción cultivada de trigo por departamento usando algoritmos de Aprendizaje no Supervizado.

El $NDVI$ es un indicador simple de biomasa fotosintéticamente activa o, en términos simples, un cálculo de la salud de la vegetación. El $NDVI$ ayuda a diferenciar la vegetación de otros tipos de cobertura del suelo (artificial) y determinar su estado general. También permite definir y visualizar áreas con vegetación en el mapa, así como detectar cambios anormales en el proceso de crecimiento. Se calcula con la siguiente expresión:

$$NDVI = \frac{NIR - Red}{NIR + Red}$$

Donde NIR es luz infrarroja cercana y Red es luz roja visible. Como se puede apreciar en esta fórmula los valores del NDVI varían de -1 a 1.

Los valores negativos indican nubes y agua, los valores positivos cercanos a cero indican suelo desnudo y los valores positivos más altos del NDVI varían desde vegetación escasa con:

$0.1 \leq NDVI \leq 0.5$ hasta vegetación verde densa $0.6 \leq NDVI \leq 1$. En general, cuando los valores del NDVI comienzan a aumentar, corresponde al inicio de la temporada de crecimiento y el período de NDVI máximo corresponde aproximadamente al final del desarrollo vegetativo o al comienzo de la etapa de floración para la mayoría de las regiones climáticas (KRIEGLER 1969).

Se dispone de datos de 291 departamentos agrícolas de la Argentina para los años desde el 2017 hasta el 2022, cada departamento tiene asociado una tabla de datos cuyas filas son series de tiempo del índice $NDVI$ y sus columnas son fechas. Cada columna representa una fecha, es decir: *'2021-06-02', '2021-06-16', '2021-07-01', '2021-07-15'* ... *'2021-12-29'*, las filas corresponden a la observación de cada serie de tiempo en un punto geográfico determinado. Las columnas por departamento y por año no tienen porque necesariamente ser las mismas. Argentina está entre los 10 países más grandes del mundo, con esto podemos suponer que la cantidad de observaciones para cada departamento puede ser considerable. Uno de los departamentos que más datos almacena del país es Rio Cuarto de la provincia de Córdoba con 1335533 filas y 535 columnas, esto lo anotamos como ejemplo para evidenciar la gran cantidad de datos que se tienen que pre-procesar y analizar teniendo en cuenta que son 291 departamentos.

La aplicación de métodos de clustering en la agrupación de cultivos en una región específica es esencial para comprender y optimizar la gestión agrícola. Un algoritmo de clustering robusto permite identificar patrones y relaciones entre diferentes tipos de cultivos, lo que facilita la toma de decisiones informadas en la planificación agrícola y la asignación de recursos. Al categorizar los cultivos según similitudes en condiciones climáticas, suelo y otros factores relevantes, se puede mejorar la eficiencia en la asignación de insumos como agua, fertilizantes y pesticidas. Además, estos algoritmos proporcionan una base sólida para la predicción de rendimientos, plagas o enfermedades específicas en función de los patrones identificados.

1.2 Procesamiento Geoespacial en la Nube.

El Cloud Computing, o computación en la nube, es un modelo de prestación de servicios tecnológicos que permite el acceso a recursos informáticos a través de Internet. En lugar de depender de una infraestructura física local, como servidores y dispositivos de almacenamiento, los usuarios pueden utilizar servicios ofrecidos por proveedores de nube, que mantienen y gestionan estos recursos en centros de datos distribuidos globalmente. Existen varias empresas que prestan estos servicios de la nube como lo son: AWS (Amazon Web Services), GCP (Google Cloud Platform), Microsoft Azure, entre otras.

Existen importantes ventajas para los usuarios que hagan uso de la computación en la Nube (Cloud Computing), podemos destacar las siguientes:

Flexibilidad: Debido a la arquitectura de la computación en la nube, las empresas y sus usuarios pueden acceder a los servicios de la nube desde cualquier lugar con una conexión a Internet para escalar servicios o reducir la escala verticalmente según sea necesario.

Eficacia: Las empresas pueden desarrollar nuevas aplicaciones y ponerlas a producir rápidamente sin preocuparse por la infraestructura subyacente, en algunos casos la adquisición de tecnología como servidores por ejemplo, puede ser más dispendioso dado que si la necesidad de procesar datos crece se deberá adquirir más servidores, si disminuye los servidores pueden

quedar obsoletos. En el caso de la nube, el consumo es a demanda haciendo más eficiente el desarrollo y producción.

Ofrece valor estratégico: Debido a que los proveedores de servicios en la nube se mantienen al tanto de las innovaciones más recientes y las ofrecen como servicios a los clientes, las empresas pueden obtener ventajas más competitivas y un retorno de la inversión más alto que si se hubiesen invertido en tecnologías que pronto serán obsoletas.

Seguridad: Las empresas suelen preguntarse cuáles son los riesgos de seguridad de la computación en la nube. Estas se consideran relativamente bajas. En general, la seguridad de la computación en la nube se reconoce como más sólida en comparación con los centros de datos empresariales, debido a la profundidad y amplitud de los mecanismos de seguridad que implementan los proveedores de servicios en la nube. Además, los equipos de seguridad de los proveedores de servicios en la nube son conocidos como los mejores expertos en el campo.

Rentabilidad: Cualquiera sea el modelo de servicio de computación en la nube que se use, las empresas solo pagan por los recursos de computación que utilicen. No necesitan sobredimensionar la capacidad del centro de datos para hacer frente a los aumentos inesperados en la demanda o al crecimiento empresarial, y pueden implementar al personal de TI para trabajar en iniciativas más estratégicas.

Computación en la Nube para el Procesamiento Geoespacial.

El procesamiento geoespacial, que implica la gestión y análisis de grandes volúmenes de datos geográficos, se ha venido migrando hacia el Cloud Computing debido a varias razones clave que aprovechan las ventajas únicas que ofrece la nube. A continuación, se explica cómo y por qué se está realizando esta migración:

1. Gestión de Grandes Volúmenes de Datos.

Almacenamiento Escalable: Los servicios en la nube ofrecen almacenamiento escalable que puede crecer según las necesidades. Esto es crucial para el procesamiento geoespacial, donde los volúmenes de datos pueden alcanzar petabytes. Los proveedores de nube, como AWS, Azure y Google Cloud, ofrecen servicios de almacenamiento masivo como Amazon S3, Azure Blob Storage y Google Cloud Storage, que permiten almacenar grandes cantidades de datos de forma eficiente y económica.

2. Costos Reducidos (si se usa el servicio eficientemente).

El almacenamiento en la nube utiliza un modelo de pago por uso, lo que significa que las organizaciones solo pagan por el almacenamiento que realmente utilizan, sin necesidad de grandes inversiones iniciales en infraestructura. Es importante remarcar que debe existir personal capacitado para el uso de los servicios de la nube, pues el uso inadecuado y poco eficiente de sus servicios puede derivar en costos que por el contrario pueden ser excesivos.

3. Capacidad de Procesamiento Potente.

Infraestructura de Alta Capacidad: Los proveedores de nube ofrecen recursos de computación de alta capacidad que pueden manejar el procesamiento intensivo de datos geoespaciales. Esto incluye el uso de máquinas virtuales (VMs) y servicios de computación como Amazon EC2, Azure Virtual Machines y Google Compute Engine. Así como el uso de tecnologías de procesamiento distribuido, como Apache Hadoop y Apache Spark, en la nube permite dividir

y procesar grandes conjuntos de datos en paralelo, lo que acelera significativamente el tiempo de análisis.

4. Acceso a Herramientas y Servicios Especializados.

Plataformas GIS en la Nube: Servicios especializados como Google Earth Engine, AWS Ground Station y Azure Maps proporcionan herramientas y plataformas específicas para el procesamiento y análisis geoespacial. Estas plataformas ofrecen capacidades avanzadas, como la visualización de datos geoespaciales, análisis de imágenes de satélite y integración de datos de sensores IoT.

5. APIs y Servicios de Machine Learning.

La integración con servicios de machine learning en la nube permite aplicar técnicas avanzadas de análisis y predicción a los datos geoespaciales. Por ejemplo, los servicios de aprendizaje automático en la nube pueden utilizarse para la clasificación de imágenes, detección de cambios y análisis predictivo.

6. Colaboración y Acceso Global.

Colaboración Remota: Los datos y resultados del análisis geoespacial pueden ser compartidos fácilmente entre equipos distribuidos globalmente a través de la nube. Esto facilita la colaboración en tiempo real y la toma de decisiones basada en datos.

7. Seguridad y Cumplimiento

Los proveedores de nube implementan robustas medidas de seguridad para proteger los datos, incluyendo cifrado, autenticación y control de acceso. Esto es vital para datos geoespaciales sensibles, como información topográfica y de infraestructura.

8. Flexibilidad y Agilidad

La infraestructura y servicios en la nube pueden ser provisionados rápidamente, lo que permite a las organizaciones implementar y escalar proyectos geoespaciales sin retrasos. La nube permite experimentar y ajustar rápidamente las configuraciones y recursos utilizados para el procesamiento geoespacial, adaptándose a cambios en los requisitos y en el volumen de datos.

Como ejemplo de muchas aplicaciones que puedan llevarse a cabo con el (Cloud Computing) se pueden destacar las siguientes:

Monitoreo Ambiental: Análisis de grandes volúmenes de datos de satélite para monitorear el cambio climático, la deforestación y la calidad del aire.

Gestión de Desastres: Procesamiento en tiempo real de datos geoespaciales para la respuesta rápida a desastres naturales, como terremotos, inundaciones y huracanes.

Agricultura de Precisión: Uso de datos geoespaciales y análisis predictivo para optimizar el uso de recursos agrícolas, mejorar el rendimiento de los cultivos y gestionar los recursos hídricos.

La migración del procesamiento geoespacial a la nube está permitiendo a las organizaciones manejar mejor los enormes volúmenes de datos, aprovechar capacidades avanzadas de procesamiento y análisis, y mejorar la eficiencia y efectividad de sus operaciones geoespaciales.

2 Preprocesamiento y Extracción de Parámetros

En el ámbito del aprendizaje automático, el preprocesamiento de datos desempeña un papel fundamental al garantizar la calidad y la idoneidad de los datos para su análisis. Este paso crucial implica la limpieza y transformación de los datos crudos, permitiendo así que los algoritmos de machine learning puedan interpretar patrones de manera más efectiva. La calidad de los resultados obtenidos está directamente vinculada a la calidad de los datos procesados, ya que problemas como valores atípicos, datos faltantes o escalas inconsistentes pueden afectar negativamente el rendimiento de los modelos. Además, el preprocesamiento facilita la interpretación y generalización de los modelos, contribuyendo a la creación de sistemas más precisos y robustos. Por lo tanto, el preprocesamiento de datos no solo es necesario, sino esencial para maximizar el rendimiento y la eficacia de los modelos de machine learning.

Por lo general en el inicio de este preprocesamiento se recurre a técnicas de reducción de la dimensionalidad cuyo empleo es imperativo en el machine learning pues la presencia de un gran número de características o variables puede generar una complejidad abrumadora en los modelos, al reducir la dimensionalidad se busca simplificar la representación de los datos sin perder información esencial, lo que conduce a modelos más eficientes y generalizables.

2.1 Valores atípicos

La presencia de valores atípicos en los datos representa un desafío significativo en el campo del aprendizaje automático, ya que puede distorsionar la interpretación y el rendimiento de los modelos. Estos valores extremos pueden afectar adversamente la precisión y la robustez de los algoritmos, ya que pueden ejercer una influencia desproporcionada en la formulación de patrones y decisiones del modelo.

La detección y gestión de valores atípicos son cruciales durante el preprocesamiento de datos, ya que estos pueden introducir sesgos y errores en la fase de entrenamiento. Los modelos pueden ser particularmente sensibles a valores atípicos, lo que puede resultar en la generación de patrones no representativos y, por ende, en predicciones menos precisas.

La detección de valores atípicos puede empezar con un simple vistazo de los datos en algunas instancias o variables, para de esta manera hacerse una idea de la problemática presente en como los datos atípicos se están manifestando.

A continuación veremos en un contexto de series de tiempo como pueden detectarse y tratarsen los valores atípicos. Estudiaremos dos propuestas robustas para estimar la media y el desvío estándar poblacionales y luego expondremos el filtro Hampel para la detección de valores atípicos en series de tiempo.

2.1.1 Filtro Hampel

En el marco de series de tiempo de datos de satélite se pueden presentar errores de medición y datos que parecen estar contaminados, esto genera un sesgo nocivo para detectar patrones intrínsecos en los datos. Quizá una manera eficaz para detectar estos valores atípicos y eliminarlos es el filtro de Hampel, cuya finalidad es la reducción de ruido la cual es un paso típico de preprocesamiento para mejorar los resultados.

Antes de la presentación formal del filtro Hampel debemos remarcar que en el contexto de la estadistica robusta (Rey 2012) la mediana de una cierta muestra es una estimación más robusta que la media muestral tradicional, este último estadístico padece la presencia de valores atípicos en datos con gran cantidad de ruido. Así mismo existen versiones robustas para la desviación estándar σ, presentaremos en este trabajo estadísticos robustos para la desviación estándar poblacional uno que se construye precisamente usando la mediana de la muestra y otro que se basa en el concepto de *trimming*.

Las definiciones de los estadísticos de la Media, los estadísticos de orden, la mediana y el desvío estándar muestrales si bien son ampliamente conocidas las damos a continuación, junto con el estadísticos robustos que se estudian en esta tesis para estimar la media y la desviación estándar poblacionales μ y σ respectivamente.

Para una muestra aleatoria $X_1, X_2, X_3, \ldots, X_n$, definimos:

- **Media muestral:**

$$\overline{X} = \frac{\sum_{i=1}^{n} X_i}{n} \tag{2.1}$$

- **Estadísticos de Orden:** Definimos $X_{(k)}$ como el estadístico de orden $k-$ ésimo de una muestra. Es decir, el $k-$ ésimo valor más pequeño de la muestra. De esta manera por ejemplo:

$$X_{(1)} = \min\{X_1, X_2, X_3, \ldots, X_n\} \tag{2.2}$$

$$X_{(n)} = \max\{X_1, X_2, X_3, \ldots, X_n\} \tag{2.3}$$

Por lo tanto, el estadístico definido en ecuación 2.2 es el minimo de la muestra y ecuación 2.3 representa el máximo de la muestra.

- **Mediana muestral:** Definimos la mediana de la muestra como *la mitad* de los datos. Más precisamente:

$$mediana(X) = \begin{cases} X_{(\frac{n+1}{2})} & \text{si } n \text{ es impar} \\ \dfrac{X_{(\frac{n}{2})} + X_{(\frac{n}{2}+1)}}{2} & \text{si } n \text{ es par} \end{cases} \tag{2.4}$$

- **Desvio estándar muestral:**

$$s = \sqrt{\frac{\sum_{i=1}^{n}(X_i - \overline{X})^2}{n-1}} \tag{2.5}$$

- **Desviación Absoluta Mediana:** Definimos la desviación Absoluta Mediana (MAD por sus siglas en inglés) de la siguiente manera:

$$MAD = mediana|X_i - mediana(X)| \tag{2.6}$$

La ecuación 2.6 se presenta como una versión robusta de la desviación en una muestra que pueda estar contaminada con valores atípicos.

En la estadística descriptiva (Fernández et al. 2002), los estadísticos mostrados en la ecuación 2.1 y la ecuación 2.4 son los ejemplos más comunes de las llamadas medidas de tendencia central, pues estas pretenden medir *el centro* de los datos estimando de esta forma la media de la población, así mismo los estimadores de la ecuación 2.5 y la ecuación 2.6 indican que tanto se dispersan los datos respecto del promedio de los mismos, es decir mide la variabilidad de la población. En la construcción de modelos probabilísticos son de preferencia estimadores cuyas medidas de dispersión sean bajas, dado que variabilidades altas de los estimadores pueden contaminar nocivamente los modelos y las predicciones (Gutmann and Hyvärinen 2010).

A modo de visualización del comportamiento de los estimadores robustos propuestos en la ecuación 2.4 y la ecuación 2.6, y los no robustos definidos en la ecuación 2.1 y la ecuación 2.5 para estimar parámetros poblacionales μ y σ, simulamos datos contaminados y no contaminados para observar la adaptación de los estimadores en los dos marcos distintos figura 2.1, interesa ver si en un contexto de datos sin valores atípicos los estimadores robustos son similares a los convencionales (no robustos) y así mismo es de interés observar si en datos con outliers los estimadores robustos logran mantener la compostura mientras sus pares no robustos pueden verse muy afectados por los datos corruptos.

Se observa en figura 2.1 para datos sin valores atípicos (gráfico izquierdo) un comportamiento similar de los estimadores robustos (en negro) y convencionales no robustos (en rojo) para la estimación de los parámetros poblacionales μ y σ, parece que para un marco de datos sin presencia de outliers podemos optar por usar cualquier pareja de estimadores. Para el caso de datos contaminados se presencia una clara corrupción en el estimador de la media $\overline{X}$ no robusto (gráfico de la derecha) pues esta estimación se corre muy hacia la derecha *creyéndole* a los outliers, así mismo para el estimador convencional de σ se obtiene un valor muy elevado en la dispersión de los datos, mientras que el MAD se adapta mejor a la verdadera naturaleza de los datos y tolera más aceptablemente los valores atípicos de la muestra.

Otra versión de estadísticos robustos para los parámetros poblacionales de interés ampliamente usadas y conocidas en la literatura son los estadísticos robustos basados en *Trimming* (Ruppert and Carroll 1980) (recorte en inglés) los cuales se contruyen ordenando los datos y eliminando un porcentaje determinado de los datos más pequeños y más grandes antes de calcular los estimadores convenciones $\overline{X}$ y s. Los estadísticos obtenidos a partir del conjunto de datos reducido (trimeado) son más robustos frente a los outliers. Existen esfuerzos en la literatura

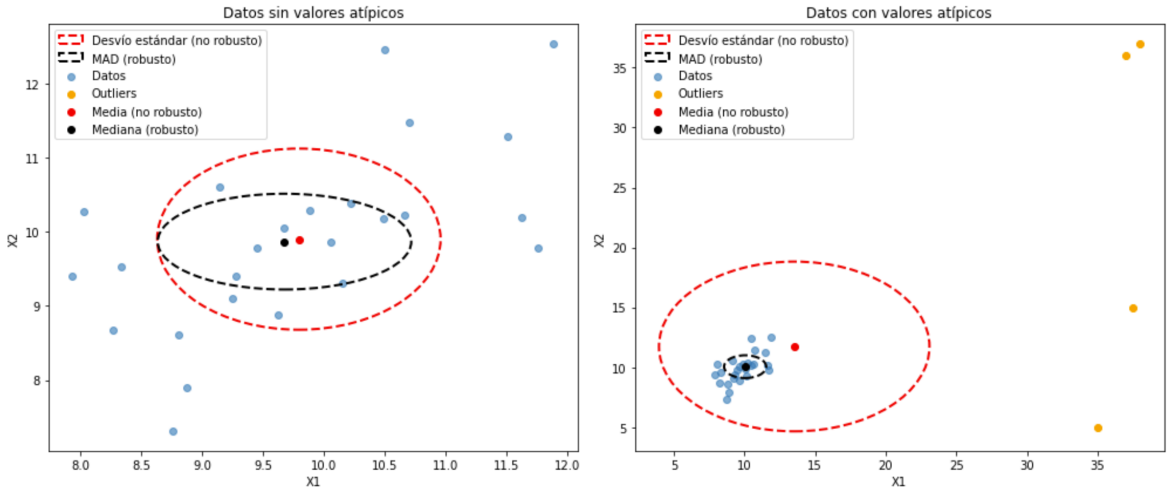

Fig 2.1: Comportamiento de los estimadores Robustos y no robustos propuestos en datos contaminados y no contaminados.

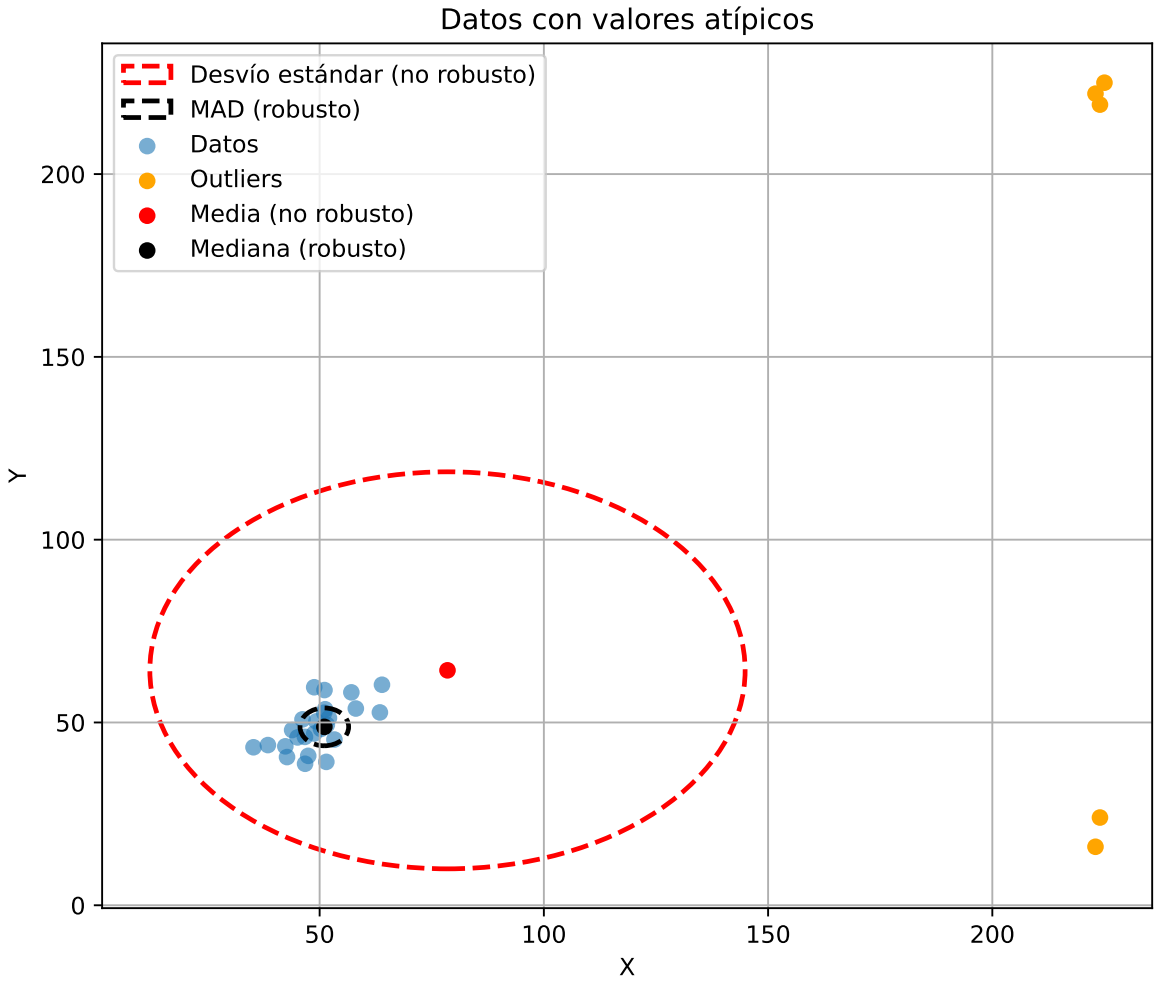

Fig 2.2: Comportamiento de los estimadores de la mediana y el MAD para datos contaminados.

para establecer cotas de error y estudiar propiedades de estadísticos trimeados para μ en (Lugosi and Mendelson 2021) y (Bickel 1965).

Por lo tanto eliminamos un porcentaje α de los datos más bajos y más altos, siendo n el tamaño de la muestra, los datos recortados serán $k = [\alpha n]$, por lo que los estimadores robustos propuestos serán.

- **Media muestral *Podada*:**

Se recortan $k/2$ observaciones más pequeñas, y la misma cantidad de las observaciones más altas:

$$\overline{X_{trim}} = \frac{\sum_{i=1+k/2}^{n-k/2} X_i}{n-k} \tag{2.7}$$

- **Desvio estándar muestral *Podada*:**

Se recortan $k/2$ observaciones más pequeñas, y la misma cantidad de las observaciones más altas:

$$s_{trim} = \sqrt{\frac{\sum_{i=1+k/2}^{n-k/2} (X_i - \overline{X_{trim}})^2}{n-k-1}} \tag{2.8}$$

Se observa un comportamiento errático de los estimadores convencionales (en rojo), mientras que los estadísticos trimeados logran adaptarse más a los datos (en negro).

Volviendo a la explicación del filtro de Hampel, este primero fija una ventana móvil para cada punto x de la muestra compuesta de K vecinos y calcula el MAD de esa ventana, si existe una observación x^* talque el MAD de esta y de sus K vecinos es mayor una cierta cantidad de desvíos, entonces este valor x^* se considera atípico y se elimina, o se hace lo que se llama en la literatura *imputación* (Donders et al. 2006) que consiste reemplazar este valor por el vecino más cercano o alguna función que dependa de sus k vecinos más cercanos, desechando de esta forma datos impuros, daremos más enfoque a este método en la sección 2.2.1.

Para datos normalmente distribuidos $X \sim \mathcal{N}(\mu, \sigma^2)$, un estimador robusto del desvío estándar poblacional σ más razonable será $\hat{\sigma} = kMAD$, donde $k \approx 1.4826$. Para demostrar esto primero tomamos:

$$Y = X - mediana(X) \tag{2.9}$$

debemos tener en cuenta que: $|Y| = |X - mediana(X)| = |X - \mu|$ para X cuya distribución es simétrica, pues la mediana coincide con la esperanza μ, por definición de la Desviación Absoluta Mediana (MAD), por ecuación 2.9 y recordando una vez más que X se distribuye simétricamente tenemos:

$$MAD = mediana|X - \mu| = mediana|Y|$$

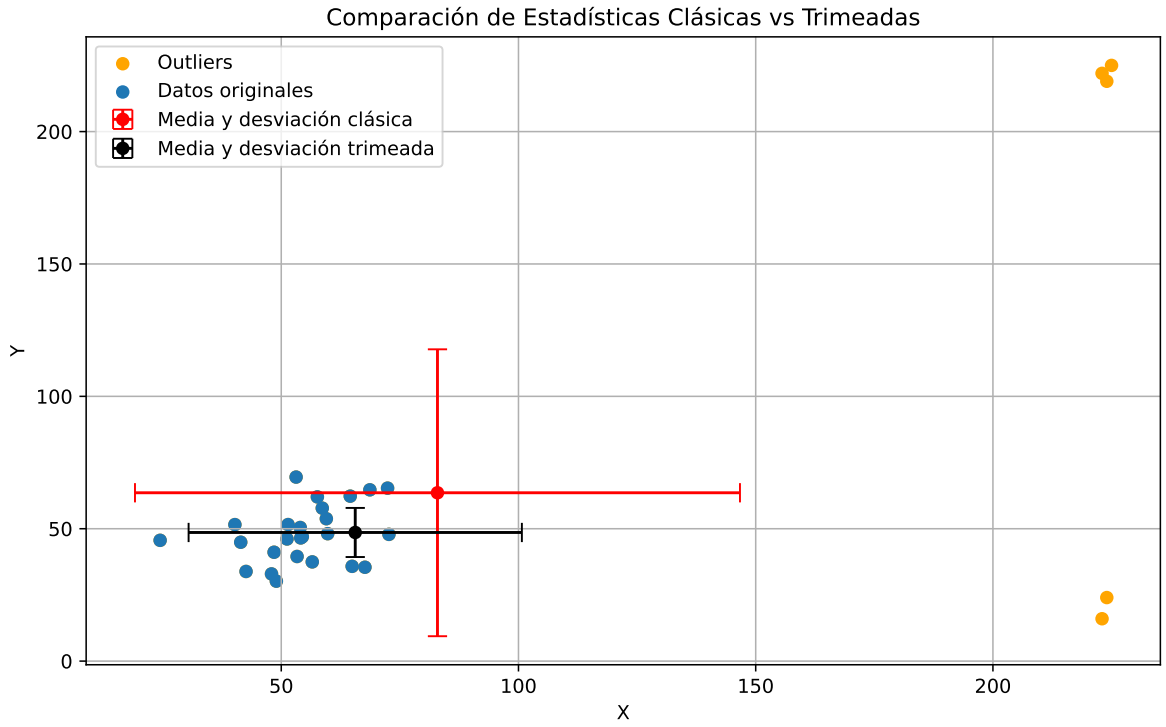

Fig 2.3: Comportamiento de los estimadores no robustos y basados en Trimming para datos contaminados, se realizó un trimming del 15 porciento.

Al calcular la mediana de $|X-\mu|$, el MAD al ser en esencia la mediana de la variable aleatoria $|Y|$, debe cumplir que el 50% de los datos están por debajo del MAD, más precisamente:

$$0.5 = P(|Y| < mediana|Y|) = P(|X-\mu| < MAD) \tag{2.10}$$

Dividiendo por σ tenemos que:

$$P\left(\frac{|X-\mu|}{\sigma} < \frac{MAD}{\sigma}\right) = 0.5$$

Obteniendo de esta forma el valor absoluto de una distribución normal estándar $Z = \frac{X-\mu}{\sigma}$, $Z \sim \mathcal{N}(0,1)$, pues $\sigma > 0$:

$$P\left(\frac{|X-\mu|}{\sigma} < \frac{MAD}{\sigma}\right) = P\left(|Z| < \frac{MAD}{\sigma}\right) = P\left(Z < \frac{MAD}{\sigma}\right) - P\left(Z < -\frac{MAD}{\sigma}\right)$$

Denotamos Φ como la función de distribución acumulada de una variable aleatoria Normal estándar ($\Phi(x) = P(Z \leq x)$), hemos obtenido:

$$\Phi\left(\frac{MAD}{\sigma}\right) - \Phi\left(-\frac{MAD}{\sigma}\right) = 0.5$$

Por simetría de la distribución normal: $\Phi(-MAD/\sigma) = 1-\Phi(MAD/\sigma)$, con lo cual resulta:

$$2\Phi\left(\frac{MAD}{\sigma}\right) = 1.5$$

Multiplicando por 0.5 y aplicando la inversa de la función Φ se obtiene:

$$\frac{MAD}{\sigma} = \Phi^{-1}(0.75)$$

Despejando σ se obtiene una estimación del mismo:

$$\hat{\sigma} = \frac{MAD}{\Phi^{-1}(0.75)}$$

Hemos demostrado que el factor k es:

$$k = \frac{1}{\Phi^{-1}(0.75)} \approx 1.4826$$

.

Así que el estimador de σ robusto y corregido con un valor de escala bajo la hipótesis de Normalidad de los datos es:

$$\hat{\sigma} = 1.4826MAD \tag{2.11}$$

con MAD definida la en ecuación 2.6. El estimador no es insesgado pero si es asintóticamente insesgado, esto es que su sesgo tiende a cero a medida que aumenta el tamaño de la muestra. Para alcanzar la insesgadez para todo n existen modificaciones en la literatura para disminuir el sesgo para muestras finitas (Akinshin 2022).

Presentamos un código en el lenguaje de programación Pyhton en el que llevamos a cabo el filtro Hampel, se pueden evidenciar los pasos que hemos expuesto anteriormente. En este código se opta por eliminar el valor atípico y dejarlo como un dato faltante, no se hace ningún tipo de imputación.

```
def hampel_filter(data, window_size, threshold):
    n = len(data)
    filtered_data = data.copy()
    for i in range(n):
        if i < window_size // 2 or i >= n - window_size // 2:
            continue
        window = data[i - window_size // 2:i + window_size // 2 + 1]
        median = np.median(window)
        mad= 1.4826*np.median(np.abs(window - median))
        if np.abs(data[i] - median) > threshold * mad:
            filtered_data[i] = np.nan
    return filtered_data
```

Aplicamos la función del filtro hampel a un ejemplo con datos simulados y vemos que realiza lo esperado figura 2.4. La serie de tiempo que se ha creado es una superposición de una función trigonométrica y una función normal. Se han agregado outliers intencionalmente con el objetivo de mostrar como el filtro hampel es capaz de detectarlos y eliminarlos.

```
window_size = 5
threshold = 2.5
serie_filtrada = hampel_filter(serie_tiempo, window_size, threshold)
```

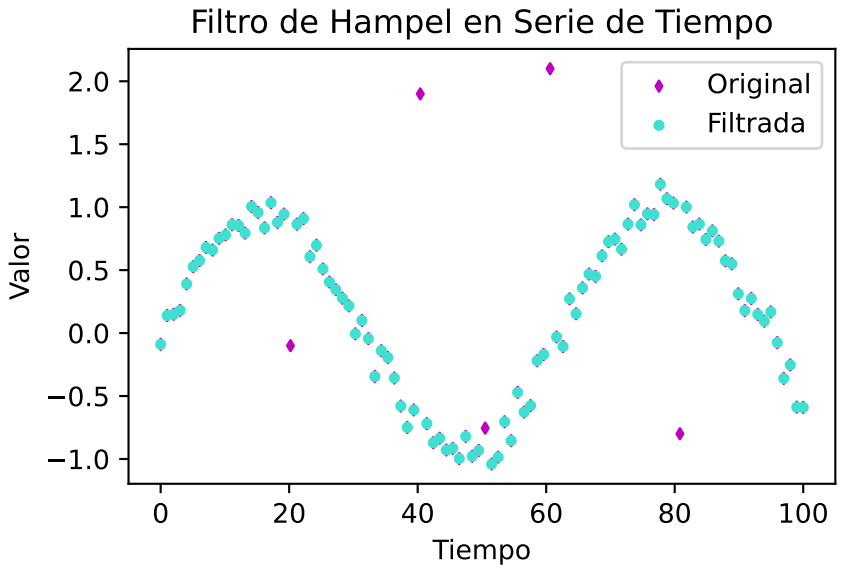

Fig 2.4: Vemos como el filtro Hampel detecta los saltos abruptos de la serie de tiempo que parecen ser valores atípicos.

Vemos como a pesar de detectar valores atípicos figura 2.4 también se eliminan datos que parecen no serlo, el filtro Hampel puede considerar valores no atípicos como atípicos.

2.2 Tratamiento de valores faltantes y Feature Extraction

En machine learning, los valores faltantes en los datos pueden generar problemas graves, como modelos sesgados, disminución en la precisión y dificultades en el análisis. Este problema ocurre cuando ciertas entradas de datos no están registradas, lo que impide que los algoritmos aprendan correctamente. Existen varias técnicas para tratar este problema:

1. Eliminación de filas o columnas incompletas, lo cual es sencillo pero puede llevar a la pérdida de información valiosa.

2. Imputación de valores faltantes, donde se sustituyen los datos perdidos por la media, mediana, moda o valores generados por modelos más avanzados como vecinos más cercanos o regresiones.

3. Uso de modelos que manejan valores faltantes de forma nativa, como algunos árboles de decisión (Khosravi et al. 2020), que pueden tratar los datos incompletos de forma más robusta, existen múltiples esfuerzos en la literatura que manejan estos problemas en distintos modelos de aprendizaje (Sued and Yohai 2013), (Danilov, Yohai, and Zamar 2012), (Sued, Valdora, and Yohai 2020).

4. Extraer de los datos un cojunto de parámetros y llevar el problema de aprendizaje a esos parámetros, en lo que se conoce como *Feature Extraction*. Alternativa que optamos llevar a cabo en esta tesis.

2.2.1 Imputación y Suavizado

Imputación

En un conjunto de datos que contiene valores faltantes, es razonable considerar reemplazar estos valores faltantes por valores numéricos teniendo en cuenta las relaciones posibles entre las observaciones vecinas (Donders et al. 2006). Esto implica hacer estimaciones de los valores perdidos y sustituirlos en los lugares donde faltan. A continuación, se describen varios métodos comunes para la imputación de datos:

- *Imputación por el vecino más cercano:*

Esta técnica reemplaza el valor faltante con el valor más cercano disponible en el conjunto de datos. El vecino más cercano se determina generalmente por la distancia en el espacio de características, y se asume que valores similares están más cerca entre sí. Este método puede ser efectivo cuando los datos presentan alta redundancia y similitud local.

- *Imputación por la media de los valores vecinos:*

Este método simple consiste en calcular la media de los valores numéricos disponibles que rodean al valor faltante y usar esta media como estimación para el valor perdido. Este enfoque asume que los valores vecinos son representativos del valor faltante y que la media es una buena estimación.

- *Imputación mediante regresión:*

En este método, se utiliza una regresión basada en los valores numéricos disponibles para estimar los valores faltantes. Primero, se desarrolla un modelo de regresión utilizando los datos completos. Luego, se aplican las fórmulas de regresión calculadas para imputar los valores faltantes. Este enfoque puede ser más preciso que la imputación por la media si la relación entre las variables está bien definida.

Es importante destacar que estos métodos de imputación requieren una eliminación previa de valores atípicos adecuada. Los valores atípicos pueden distorsionar las imputaciones si no se manejan correctamente, ya que la confianza en los datos vecinos implica que los valores cercanos sean representativos y no estén influenciados por outliers. Si bien la imputación de valores faltantes puede resolver el problema de los datos incompletos, no tratar adecuadamente los valores atípicos puede introducir nuevos problemas y sesgos en el análisis posterior. Por lo tanto, es crucial realizar una limpieza de datos exhaustiva antes de proceder con la imputación.

La imputación en general más aún la creada por el método de regresión lineal para una gran cantidad de valores faltantes puede ser un proceso computacionalmente muy costoso tanto en tiempo como en memoria requerida, además de esto no reduce la dimensionalidad, por lo que si bien ha sido explorado como alternativa escogemos no llevarlo a cabo para la solución del problema, debido a que la eficiencia también juega un factor fundamental en este asunto debido a la gigantesca cantidad de series de tiempo a analizar.

Suavizado

La herramienta Suavizado de serie temporal suaviza una variable numérica de una o varias series temporales mediante promedios móviles centrados, hacia delante y hacia atrás, así como un método adaptable basado en regresión lineal local.

Las técnicas de suavizado de series temporales se utilizan ampliamente en múltiples campos como por ejemplo en, procesamiento de señales y otros campos que manejan datos recopilados a lo largo del tiempo. El suavizado de datos temporales a menudo revela tendencias o ciclos a largo plazo mientras suaviza el ruido y las fluctuaciones a corto plazo.

El suavizado de series temporales es aplicable a cualquier serie temporal que se sepa que contienen ruido o fluctuaciones a corto plazo. Por ejemplo, en nuestro caso el índice NDVI puede tener fluctuaciones abruptas de una observación a otras en corto tiempo debido a cualquier inconveniente como errores del satélite, alguna nube que interfiera o cualquier fenómeno que no podemos controlar. A continuación aplicamos el método de suavizado usando el filtro de Savitzky–Golay (Savitzky and Golay 1964)

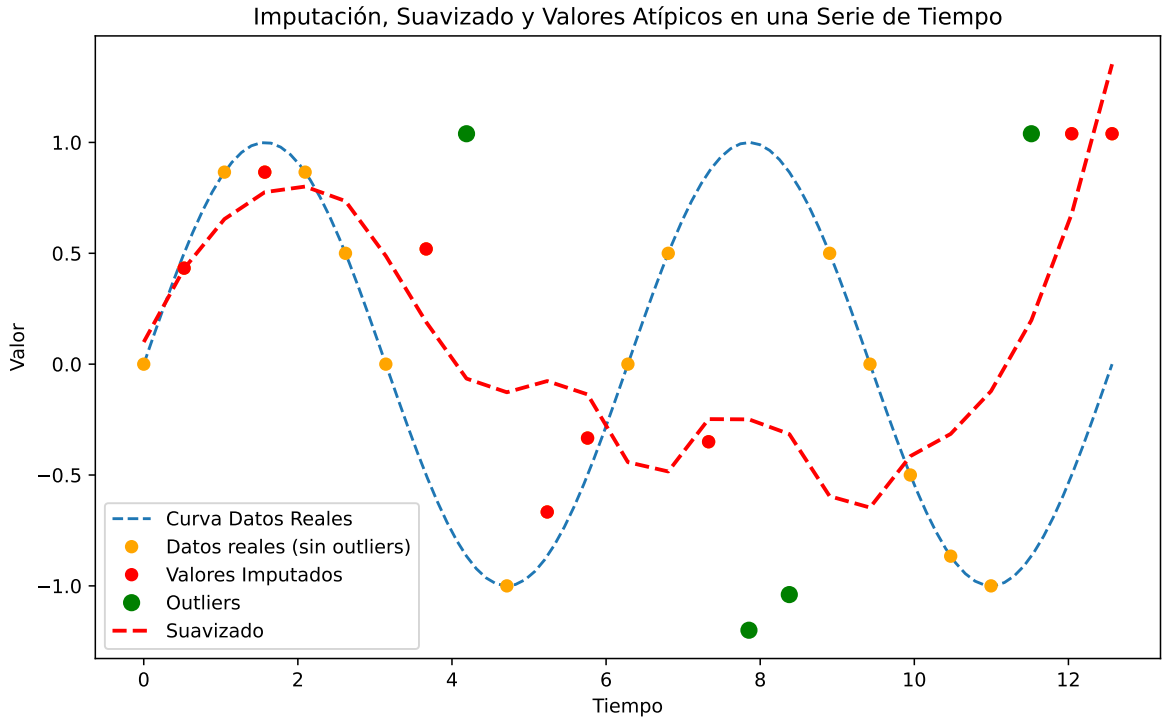

Fig 2.5: Vemos como los outliers en verde sesgan nocivamente la imputación, generando un suavizado defectuoso.

El filtro de Savitzky–Golay se basa en el cálculo de una regresión polinomial local de grado k para cada ventana que se va tomando. Los valores de k y el ancho de ventana son parámetros del modelo que para el gráfico generado en figura 2.5 tomamos igual como 3 y $L = 11$, donde L es el ancho de ventana. No pretendemos en este trabajo hacer un estudio exhaustivo del método de suavizado del filtro de Savitzky–Golay, pero si lo presentamos porque se consideró como una opción a seguir para suavizar las series de tiempo e intentar robustecerlas, aunque no se terminó optando por este camino debido a que no existe una reducción de la dimensionalidad en este procedimiento y un posterior proceso de clustering podría ser computacionalmente demasiado costoso, para conjuntos de datos voluminosos, y para más de 200 departamentos. Adicionalmente vemos en figura 2.5 una pobre adaptación a valores atípicos groseros, dándonos poca robustez, si bien el previo tratamiento del filtro Hampel elimina gran parte de los valores atípicos en series de tiempo no se garantiza la perfección en la limpieza para las múltiples series de tiempo analizadas.

2.2.2 Extracción de Features

Extracción de Features se define como un conjunto de métodos con los que se busca transformar los datos originales a un espacio de menor dimensionalidad, intentando extraer la mayor cantidad de información y almacenando la esencia del conjunto de datos original sin perder información relevante del mismo. Es como destilar los elementos esenciales de un conjunto de datos, ayudando a simplificar y resaltar los aspectos clave mientras se filtran los detalles menos significativos. Es una forma de centrarse en lo que realmente importa de los datos.

Existen varios métodos conocidos como el Análisis de Componentes Principales (PCA) que se enmarca en un contexto de reducción de la dimensionalidad. En análisis de imágenes, decodificación de texto la extracción de features es de gran relevancia (Daffertshofer et al. 2004).

En este estudio extraeremos de cada serie de tiempo un conjunto de 8 parámetros que almacenarán la esencia de la misma, esto nos permite reducir la dimensionalidad extrayendo información importante del dataframe como lo definimos en Extracción de Features. La cantidad de parámetros (igual a 8) se toma *manualmente*, se pretende tomar de cada serie de tiempo (que suele tener 13 columnas o más) un conjunto de parámetros menor a 13 (pues procuramos reducir la dimensionalidad, idealmente la cantidad de parámetros que extraemos debe ser menor al número de columnas). Aunque también buscamos explicar y describir la serie de tiempo (por lo que escoger una cantidad pequeña de parámetros, puede arrojarnos descripciones deficientes). Para este estudio daremos especial importancia a fiteos que logren ser resistentes al fuerte ruido que pueda presentarse.

Se plantean dos métodos de extración de parámetros, que presentamos a continuación.

2.2.3 Smooth Splines

Este problema se enmarca dentro de los problemas de interpolación. La tarea es construir una curva suave, $g(x)$, que se aproxima a los datos de entrada sin la necesidad de pasar exactamente por cada punto.

Se busca encontrar una función spline g que satisface:

$$\sum_{j=1}^{N}[w_j(g(x_j) - y_j)]^2 \leq s, \tag{2.12}$$

donde los datos están fijos y dados por los pares (x_j, y_j), N es la cantidad de datos que se quieren interpolar, las ponderaciones w_j para cada observación (x_j, y_j) son proporcionados por el usuario. Para ello, *scipy.interpolate* permite construir los smoothing splines, basados en la biblioteca Fortran FITPACK de P. Dierckx. (Dierckx 1975).

El parámetro que controla el grado de suavidad de la curva resultante $g(x)$ y que tan aproximada se encuentra la curva de los datos es s. Es decir para $s = 0$ corresponde al problema de interpolación donde $g(x_j) = y_j$, aumentar s creará curvas spline mas suaves, pues mayor es la tolerancia que existe en la diferencia $g(x_j) - y_j$.

Encontrar la mejor estimación del parámetro de suavizado s es un intento de prueba y error, si todos los pesos w_j son iguales a 1, una opción razonable podría ser $s \approx m\hat{\sigma}^2$, donde $\hat{\sigma}$ es un estimador del desvío estándar (Dierckx 1975).

Se proporciona el código en Python que crea el Smooth Spline para un conjunto de datos, es importante notar que se consideran sólamente los datos cuya imagen, (es decir su segunda coordenada) no es un valor faltante, pues como hipótesis fundamental los datos reales a interpolar (x_j, y_j) deben estar bien definidos. También se debe tener en cuenta que se usa la estimación de la desviación estándar recomendada: $\hat{s} = m\hat{\sigma}^2$.

```
def fit_smooth_spline_s_std_robust(row):
    x = np.arange(len(row))
    y = row.values
    mask = pd.notnull(y)
    x_no_nan = x[mask]
    y_no_nan = y[mask]
    sd_robust = len(x_no_nan)*robust_std(y_no_nan)**2
    tck = splrep(x_no_nan, y_no_nan, s = sd_robust)
    spline_smooth = BSpline(*tck, extrapolate=False)
    return spline_smooth(x)
```

A modo de ilustración veremos como se comporta el fiteo Smooth Splines para una serie de tiempo con datos espurios figura 2.6 y para datos limpios figura 2.7. Haciendo uso de distintos valores para el parámetro de suavizado s.

Presentamos una serie de tiempo que corresponde a datos reales del índice NDVI para el departamento Adolfo Alsina de la provincia de Buenos Aires, esta serie de tiempo como observamos no presenta un tratamiento de valores atípicos previo, por lo que se observan cambios abruptos del índice NDVI que parecen ser errores de medición o algún otro problema figura 2.6. Para los distintos valores de s vemos los resultados obtenidos, para s chico el fiteo padece la presencia de valores atípicos, y para el parámetro recomendado $s = n\hat{\sigma}^2$ la suavidad es mayor pero no se logra explicar la naturaleza de la serie de tiempo muy satisfactoriamente como se presencia en la figura 2.6, por lo que está presente siempre este *trade-off* entre tener una sobre-explicación de la serie de tiempo arriesgándose a sufrir el ruido nocivo que pueda manifestarse o solventar este problema aumentando el valor de s dando una explicación quizá

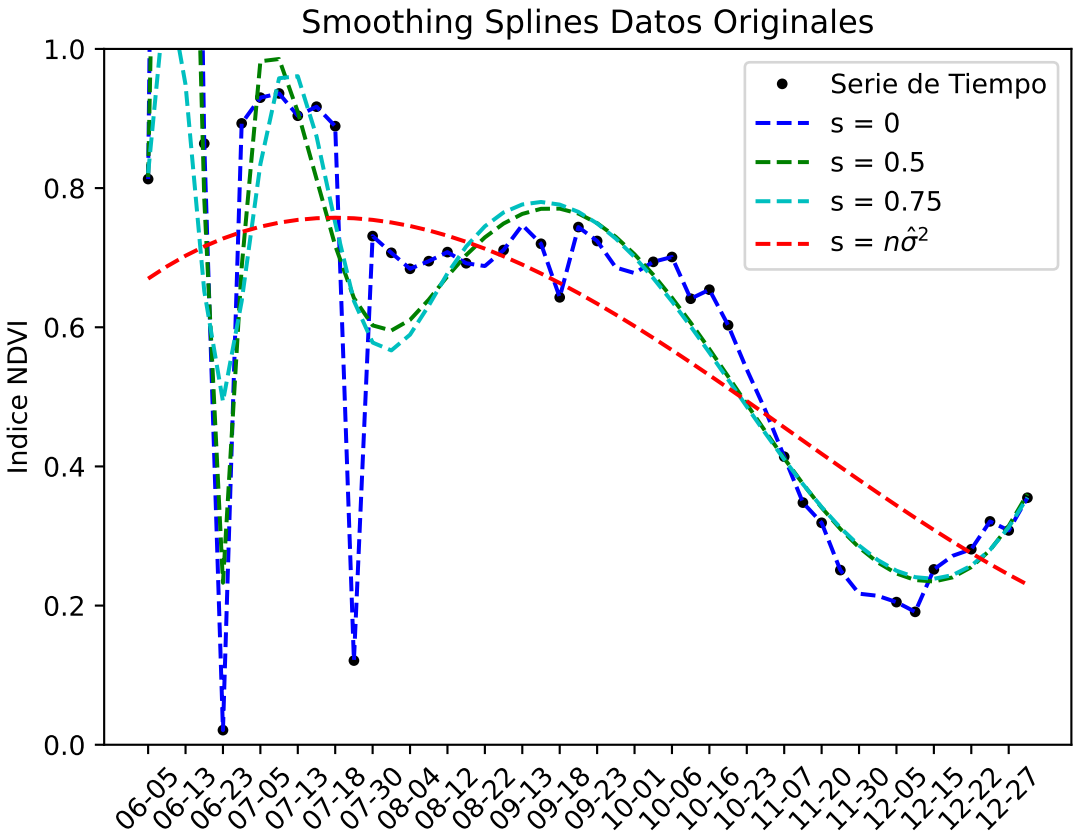

Fig 2.6: Fiteos Smooth Splines para una serie de tiempo con valores atípicos y distintos parámetros de suavizado. No se observa un fiteo aceptable para ninguno de los parámetros usados.

mas pobre de la serie de tiempo. El estimador usado para el desvío estándar poblacional es el presentado en la ecuación 2.11.

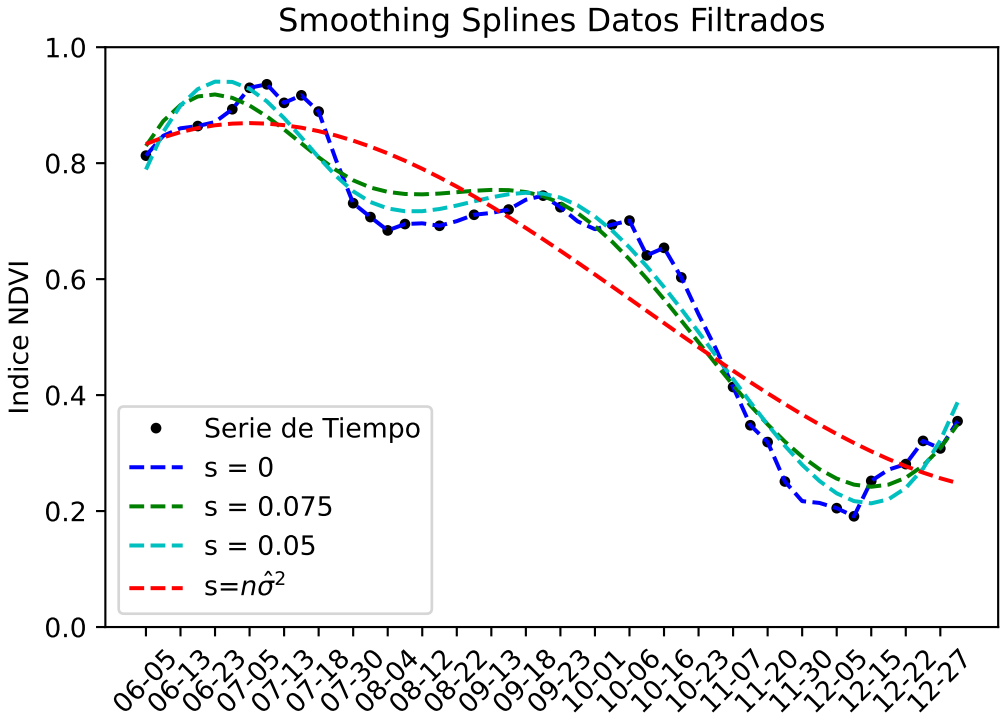

Fig 2.7: Fiteos Smooth Splines para una serie de tiempo Filtrada mediante Hampel y distintos parámetros de suavizado.

Vemos como para $s = 0$ se obtiene un fiteo perfecto figura 2.7, para datos previamente filtrados parecer ser conveniente usar el fiteo de smooth splines con $s = 0$ (o valores cercanos a cero) pues bajo este panorama de datos fiables podemos tener alta o *perfecta confianza*, almacenar la información del fiteo en 8 parámetros reduciendo de esta forma la dimensionalidad y proseguir con el estudio, pero la calidad de este proceso depende fuertemente de que tan buena sea la performance del tratamiento de datos faltantes, es decir requerimos que el filtro Hampel y otros tratamientos previamente hechos tengan una efectividad del 100% en el proceso lo cual no se puede garantizar, si se escapan valores atípicos no detectados por el filtro Hampel por ejemplo obteniendo series de tiempo defectuosas, al realizar el fiteo con $s = 0$ se pueden obtener una gran cantidad de fiteos contaminados propagando un error nocivo en futuros algoritmos de parendizaje, recordando que debemos tratar millones de series de tiempo no consideramos conveniente arriesgarnos a crear series de tiempo que propaguen ruido en el futuro estudio de aprendizaje no supervizado. Así mismo Smooth Splines no necesariamente tiene en cuenta las restricciones de f y siendo este un problema con condiciones de contorno $0 \leq NDVI \leq 1$ (por lo expuesto en la sección 1.1, sólo nos interesará los índices NDVI mayores o iguales a cero dado que los valores negativos indican nubes y agua), pueden crearse fiteos que no respeten estas condiciones y no expliquen los datos satisfactoriamente.

2.2.4 Ajuste Splines mediante optimización con restricciones.

En este sección proponemos un método alternativo que se basa en ajustar un Spline cúbico mediante técnicas de minimización con restricciones. Usando el método TRF explicado en Chapter 5 (Trust Region reflective) pretendemos encontrar:

$$\arg\min_{P\in\mathbb{R}^8} g(P). \tag{2.13}$$

Donde $g : \mathbb{R}^8 \to \mathbb{R}$ esta definida por:

$$g(P) = S(P)[t_j] - y_j.$$

Donde S es un spline cúbico definido en $\mathbb{R}$ (los splines cúbicos son funciones suaves, cumpliendo así una hipótesis fundamental del método TRF), t_j es la variable temporal sobre la cual está definida el Spline, y_j representa los datos reales del índice NDVI. Y P es el vector de 8 parámetros sobre el cual estamos minimizando, la restricción es que cada coordenada del vector P está entre cero y uno.

Se proporciona un código en lenguaje Python donde se puede evidenciar la propuesta de extracción de parámetros de esta sección, a este método lo apodamos: Splines-TRF. Es importante notar que se consideran sólamente los datos cuya imagen, (es decir su segunda coordenada) no es un valor faltante, pues como hipótesis fundamental los datos reales a interpolar (x_j, y_j) deben estar bien definidos.

```
def spline_n_params(x, p1, p2, p3, p4, p5, p6, p7, p8):
    t = np.linspace(0, 70, 8)
    p = CubicSpline(t, [p1, p2, p3, p4, p5, p6, p7, p8],
                        bc_type="natural")
    return p(x)

def fit_spline_trf(row):
    x = np.arange(len(row))
    y = row.values
    mask = pd.notnull(y)
    x_no_nan = x[mask]
    y_no_nan = y[mask]
    p0 = [0.5] * 8
    bounds = (0, [1.] * 8)
    try:
        popt, pcov = curve_fit(spline_n_params, x_no_nan,
                               y_no_nan, p0=p0, bounds=bounds,
                               method="trf")
        return popt
    except:
        return np.nan * np.ones(len(p0))
```

Como podemos observar se crea primero un Spline cúbico y después se minimiza sobre la variable $P = (p_1, p_2 \ldots, p_7, p_8)$. De la función *curve_fit* se usa el método *trf*, pues es el método

que hemos escogido como se especificó en ecuación 2.13. Las restricciones están dadas por: $0 \leq p_i \leq 1$ con $i \in \{1, \ldots, 8\}$, como se había definido en ecuación 2.13. Inicializamos todos los parámetros p_i en 0.5.

Ahora observemos el comportamiento de este procedimiento en una serie de tiempo de datos reales. Se muestra una aceptable tolerancia bajo valores atípicos del fiteo Splines-TRF representado en rojo, para los datos filtrados se ve una satisfactoria descripción de la naturaleza de la serie de tiempo figura 2.8.

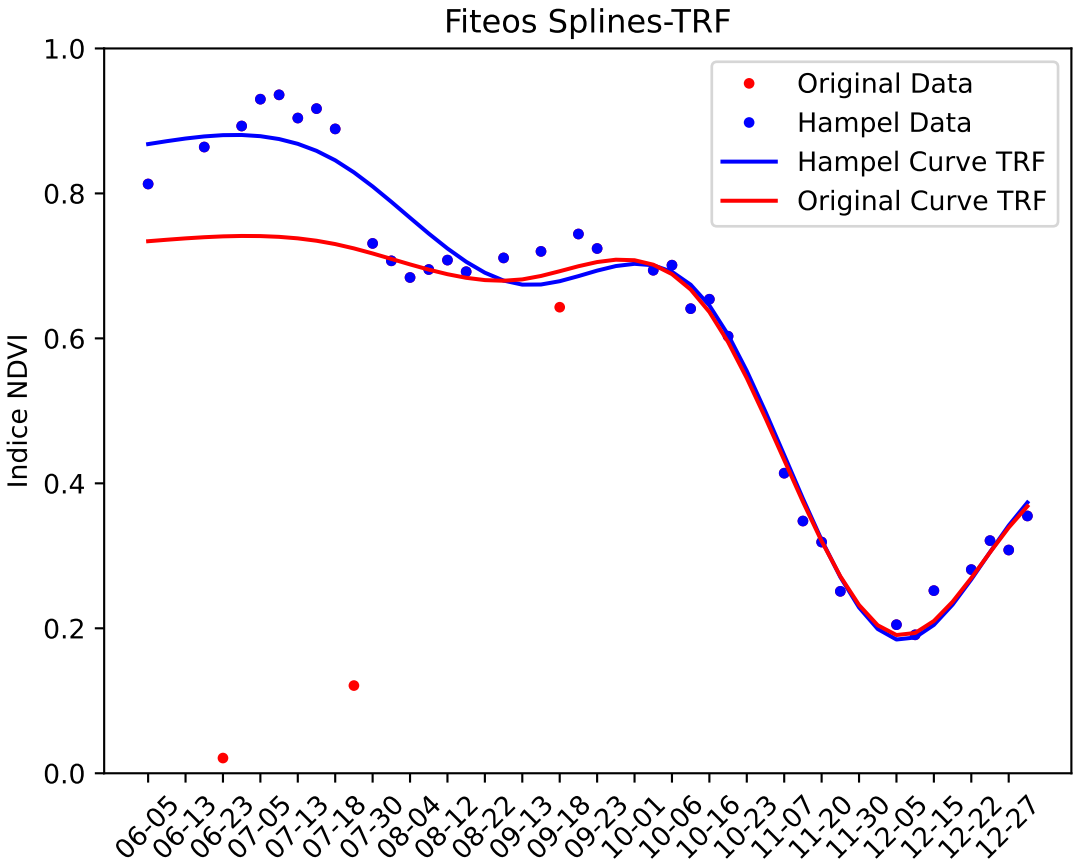

Fig 2.8: Splines-TRF para datos filtrados con Hampel y datos Originales con outliers.

Es importante notar que claramente Splines-TRF es más robusto que Smooth Splines, además el único parámetro libre de Splines-TRF es la cantidad de nodos que pretendemos extraer, en nuestro caso es igual a 8. En comparación al utilizar Smooth splines como método de extracción de parámetros también tendremos que tomar la decisión sobre cuantos parámetros extraer, además de tener que escoger el parámetro de suavidad s, El método presentado (Splines-TRF) no requiere este parámetro.

3 Clustering

3.1 Definición

Clustering es un algoritmo de Machine Learning cuya finalidad principal es lograr el agrupamiento de conjuntos de objetos no etiquetados (lo que lo hace un algoritmo no supervisado), para así construir subconjuntos de los datos conocidos como *clusters*. Cada cluster está formado por una colección de elementos que a términos de análisis resultan similares entre sí, pero que poseen características diferenciales con respecto a objetos de otros clusters.

Cabe diferenciar antes de hacer un desarrollo teórico formal, entre los problemas de clasificación y de clusterización. Ambos algoritmos moralmente presentan ciertas similitudes prácticas pero su diferencia esencial radica en que la clasificación se sirve de clases predefinidas, de las que ya a priori conocemos la respuesta, por el contrario la clusterización identifica similitudes entre objetos que agrupa según esas características en común y que les diferencian de los otros grupos de objetos. Por esta razón enmarcamos el problema de clasificación como un algoritmo Supervisado, y la clusterización como un problema No Supervisado, hecha esta aclaración conceptual prosigamos con la exposición del clustering.

En primer lugar se debe definir el número de agrupaciones que se desea hacer en el conjunto de datos, esto se puede llevar a cabo por el conocimiento previo del problema que deseamos abordar, si se dispone de una experiencia del comportamiento de los fenómenos podemos establecer una cantidad de agrupaciones apropiada. En caso contrario de no tener un entendimiento anticipado de la naturaleza del problema real podemos usar método del codo (que es una heurística muy usada) varios métodos matemáticos (Masud et al. 2018) para encontrar un número óptimo de clusters, algunos con más formalidad que otros pero siempre buscando el objetivo de establecer que cantidad de agrupaciones es la más apropiada. De ahora en más, supondremos que la cantidad de clusters es un parámetro fijo en los modelos de clustering que se presentarán en este trabajo, y no abordaremos el problema particular de encontrar el número óptimo de agrupaciones.

Existen varios métodos de clustering como lo pueden ser: k-means, Gaussian Mixture Model, hierarchical clustering, spectral clustering, DBSCAN entre otros. Haremos enfoque en los dos primeros métodos y como están relacionados entre sí. Asi como expondremos métodos de clustering robustos más adelante. Los métodos difieren por como definimos los valores iniciales, que distribuciones subyacentes usamos, y que teoría matemática desplegamos para clusterizar.

3.2 Métodos de Clusterización

Estudiaremos y presentaremos dos métodos de clusterización muy importantes k-means y Gaussian Mixture Model (GMM), veremos la relación entre estos dos métodos y observaremos

su comportamiento en datos simulados primero sin valores atípicos y con valores atípicos.

Comenzaremos con el método quizá más elemental de la teoría de clusterización. K-means es el método que menor desarrollo matemático exige, pero aún así se presenta como una buena introducción a la clusterización en general.

3.2.1 K-Means

Tenemos un conjunto de elementos $\{x_1, x_2, ..., x_N\}$, consistente de N observaciones de una variable euclidiana aleatoria D-dimensional x. particionaremos los datos en K clusters, donde supondremos por ahora que K es un valor dado. Definamos un conjunto D-dimensional de vectores μ_k (centroides), donde: $k = 1, ..., K$, en los cuales μ_k será el representante asociado para el $k-$ ésimo cluster. Más adelante veremos que estos μ_k representan los centros de cada grupo, nuestro objetivo será asignar cada dato a una agrupación, así como también encontrar un conjunto de vectores $\{\mu_k\}$, tal que la suma de los cuadrados de las distancias de cada punto de datos a su vector más cercano μ_k, sea un mínimo.

Para cada punto x_n definimos variables $r_{nk} \in \{0, 1\}$ de la siguiente manera:

$$r_{nk} = \begin{cases} 1 & \text{si } x_n \text{ es asignado al cluster } k \\ 0 & \text{en caso contrario.} \end{cases} \tag{3.1}$$

Definimos la función objetivo como:

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|x_n - \mu_k\|^2, \tag{3.2}$$

que representa la suma de las distancias de todos los puntos a su representante asignado, asi que nuestra tarea consiste en encontrar los valores r_{nk} y μ_k que minimicen J.

Esto lo llevamos a cabo iterativamente con los siguientes pasos:

1. *Inicializamos* μ_k: Se inicializan los centroides aleatoriamente.
2. *Minimizamos* J *con respecto de* r_{nk} *manteniendo* μ_k *fijo.*: Asignar las observaciones a los centroides más cercanos según la distancia L_2.
3. *Minimizamos* J *con respecto a* μ_k *manteniendo* r_{nk} *fijo.*: Se calculan los centroides promediando los puntos asignados a cada cluster.
4. *Repetimos los pasos 2 y 3*: Hasta alcanzar la convergencia o alcanzar un máximo número de iteraciones

Minimizar con respecto de r_{nk} (Paso 2) es simplemente asignar a cada x_n el μ_k más cercano, con lo que podemos expresarlo de la siguiente manera:

$$r_{nk} = \begin{cases} 1 & \text{si } k = argmin_j \|x_n - \mu_j\|^2 \\ 0 & \text{en caso contrario.} \end{cases} \tag{3.3}$$

Ahora para optimizar con respecto de μ_k con las asignaciones fijas, derivamos J respecto de los centroides e igualamos a cero, obteniendo:

$$2\sum_{n=1}^{N} r_{nk}(x_n - \mu_k) = 0.$$

Despejando μ_k (paso 3) tenemos:

$$\mu_k = \frac{\sum_{n=1}^{N} r_{nk}x_n}{\sum_{n=1}^{N} r_{nk}}, \tag{3.4}$$

donde podemos interpretar el denominador como el número de puntos x_n asignados al cluster k. Asi que μ_k no es nada menos que la media de todos los puntos x_n asignados al cluster k.

Estos dos pasos iterativos se repiten hasta que no existan reasignasiones nuevas o hasta el máximo número de iteraciones. La convergencia esta asegurada pues en cada fase se va reduciendo el valor de la función objetivo. Más adelante veremos en que caso este método puede darnos resultados no deseados.

```
from scipy.spatial.distance import cdist
def kmeans(data,K, no_of_iterations):
    idx = np.random.choice(len(data), K, replace=False)
    centroids = data.iloc[idx]
    distances = cdist(data, centroids ,'euclidean')
    points = np.array([np.argmin(i) for i in distances])
    for _ in range(no_of_iterations):
        centroids = []
        for idx in range(K):
            temp_cent = data[points==idx].mean(axis=0)
            centroids.append(temp_cent)

        centroids = np.vstack(centroids)
        distances = cdist(data, centroids ,'euclidean')
        points = np.array([np.argmin(i) for i in distances])

    return points
```

Vemos como en figura 3.1 K-means agrupa razonablemente los datos en tres clusters, esto ha sido usando el código proporcionado arriba.

3.2.1.1 Cuando K-means puede fallar?

Veremos en que casos este algoritmo que parece ser bastante razonable puede darnos aglomeraciones incorrectas. Como se puede apreciar en el método K-means solo estamos estimando los centroides de cada aglomeración, es decir no se estima la matriz de covarianzas de cada cluster.

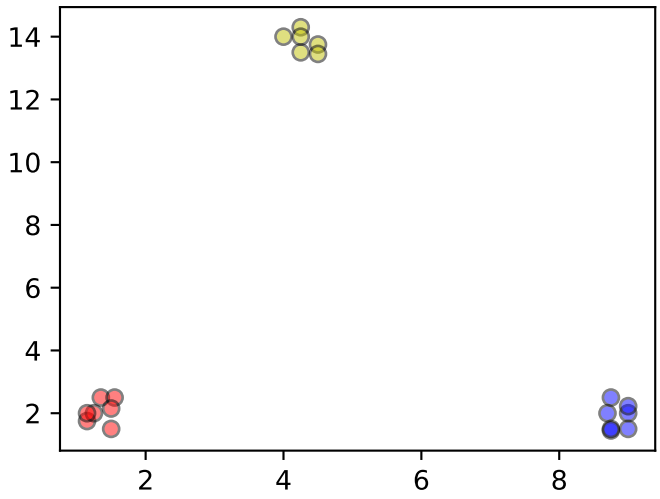

Fig 3.1: K-means, usando el código proporcionado arriba.

Por lo que aglomeraciones con varianza muy distinta, agrupaciones *deformes* o datos irregulares puede darnos predicciones erróneas. Veremos usando datos simulados las situaciones mencionadas anteriormente.

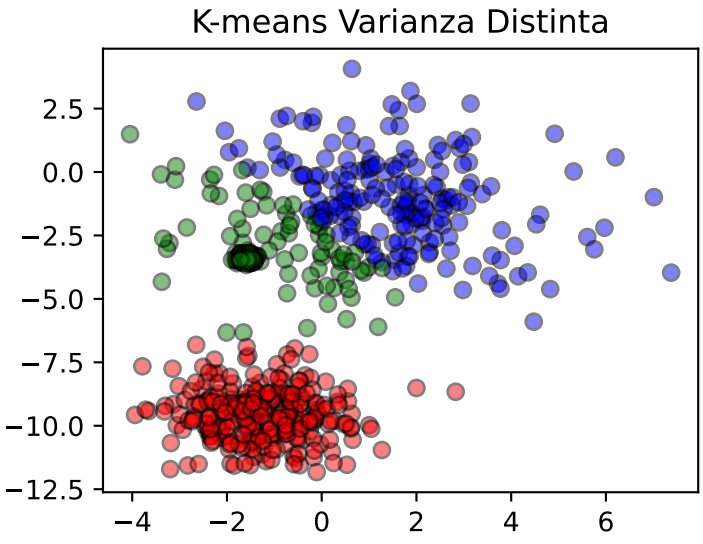

Fig 3.2: K-means con tres clusters para varianzas muy distintas.

Se ha logrado agrupar un cluster satisfactoriamente, pero la aglomeración de puntos demasiado concentrados no es identificada como cluster, pues el algoritmo no tiene en consideración estos comportamientos figura 3.2.

Vemos como K-means no logra distinguir la forma de los clusters y simplemente tiene a consideración la distancia entre los datos figura 3.3. Existen patrones que el algoritmo no puede detectar y será necesario para un modelo más adecuado tener en cuenta la covarianza

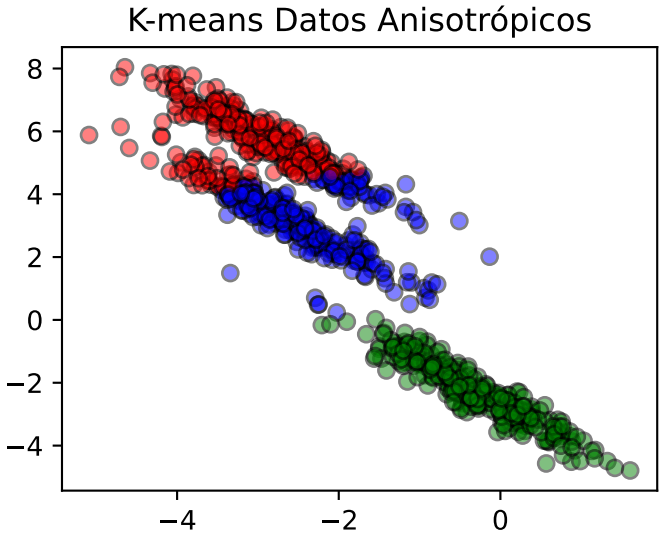

Fig 3.3: K-means con tres clusters para datos anisotrópicos.

de las agrupaciones. Se observa en figura 3.4 como K-means no logra solucionar este problema de datos irregulares.

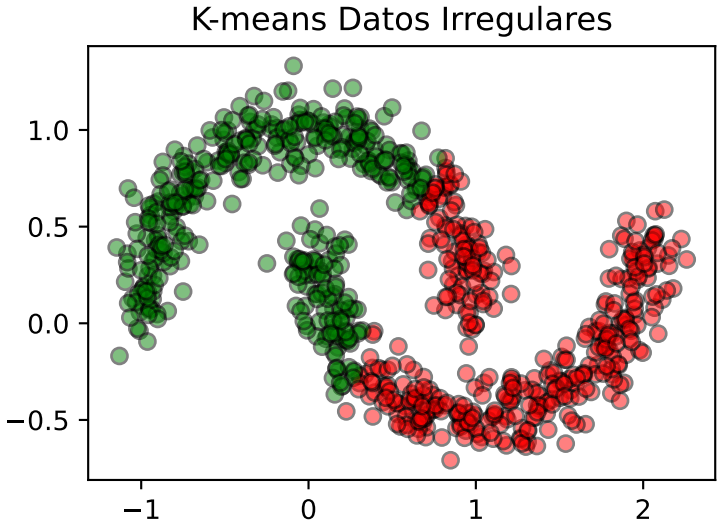

Fig 3.4: K-means no logra detectar patrones irregulares.

Presentaremos un modelo ampliamente usado para clusterizar que generaliza K-means, y soluciona varios problemas que tienen que ver con la *forma* de los clusters.

3.2.2 Gaussian Mixture Model

Para introducir este modelo que de ahora en más llamaremos GMM es ideal introducir algunos conceptos y definiciones. Para lo cual seguiremos escencialmente el (Bishop 2006).

- **Definición: (Variable Latente)**

Una variable latente es una variable para la cual no hay realización de muestra para al menos algunas observaciones en una muestra dada (Bollen 2002). En nuestro marco de clusterización esta variable latente sería la asignación de conglomerados z. Esta variable de asignación no es observada en el modelo. Si por el contrario se observa x, que son los datos en sí.

- **Definición: (Modelo de variable Latente)**

Es un modelo de probabilidad para el cual ciertas variables son latentes. Por ejemplo El GMM es un modelo de variable latente, pues en este modelo la asignación de conglomerados z es una variable que se observa.

- **Definición (*Mixture Distribution*)**

Una densidad $p(x)$ representa una *mixture distribution* o un mixture model, si podemos escribir su densidad como una combinación convexa de otras densidades. Esto es,

$$p(x) = \sum_{i=1}^{k} w_i p_i(x),$$

donde $w_i > 0$. $\sum_{i=1}^{k} w_i = 1$, y cada p_i es una densidad de probabilidad.

En nuestro GMM se supone como hipótesis fundamental que los datos x tienen una Mixture distribution cuyas densidades son normales. Es decir:

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \mathbf{\Sigma_k}). \tag{3.5}$$

La optimización de la función de verosimilitud se lleva a cabo por el algoritmo de Expectation-Maximization (EM). Daremos las siguientes definiciones que nos permitirán formalizar el problema.

Introduzcamos una variable aleatoria binaria $\mathbf{z}$ de K dimensiones: $\mathbf{z} = (z_1, z_2, .., z_n)$, talque: $p(z_k = 1) = \pi_k$,

Donde los parámetros π_k deben satisfacer: $0 \leq \pi_k \leq 1$, y además $\sum_{k=1}^{K} \pi_k = 1$

π_k representa la probabilidad de que una cierta observación pertenezca a una agrupación k-ésima. Dado que z usa una representación vectorial, podemos escribir esta distribución de la siguiente manera:

$$p(\mathbf{z}) = \prod_{k=1}^{K} \pi_k^{z_k}.$$

La distribución condicional de x dado un z es una distribución Gaussiana.

$$p(\mathbf{x}|z_k = 1) = \mathcal{N}(x|\mu_k, \mathbf{\Sigma_k}),$$

con lo cual tenemos:

$$p(\mathbf{x} \mid \mathbf{z}) = \prod_{k=1}^{K} \mathcal{N}(\mathbf{x} \mid \mu_k, \mathbf{\Sigma_k})^{z_k}.$$

Una cantidad que juega un papel fundamental es la probabilidad condicional de z dado x. La cual podemos encontrar usando el Teorema de Bayes.

$$\gamma(z_k) = p(z_k = 1|\mathbf{x}) = \frac{p(z_k = 1)p(\mathbf{x}|z_k = 1)}{\sum_{j=1}^{K} p(z_j = 1)p(\mathbf{x}|z_j = 1)} = \frac{\pi_k \mathcal{N}(x|\mu_\mathbf{k}, \mathbf{\Sigma_k})}{\sum_{j=1}^{K} \pi_k \mathcal{N}(x|\mu_\mathbf{j}, \mathbf{\Sigma_j})} \tag{3.6}$$

El problema de aprendizaje lo podemos formular de la siguiente manera:

Disponemos de un conjunto $x_1, x_2, ..., x_n$ extraído de un GMM. nuestro objetivo será estimar los siguientes parámetros:

- Probabilidades de los cluster: $\pi = (\pi_1, ..., \pi_k)$
- Centroides de los cluster: $\mu = (\mu_1, ..., \mu_k)$
- Las matrices de covarianza de cada cluster: $\mathbf{\Sigma} = (\Sigma_1, ..., \Sigma_k)$

Expectation Maximization para Mezclas Gaussianas

Un método elegante y poderoso para encontrar soluciones de máxima verosimilitud para modelos con variables latentes es el algoritmo de Expetation-Maximization, o algoritmo EM ampliamente usado en diferentes modelos. En particular, lo vamos a tratar para este caso de mezclas gaussianas. A partir de la ecuación 3.5, el logaritmo de la función de verosimilitud esta dado por:

$$Inp(\mathbf{X}|\pi, \mu, \mathbf{\Sigma}) = \sum_{n=1}^{N} In \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x_n}|\mu_\mathbf{k}, \mathbf{\Sigma_k}) \right\} \tag{3.7}$$

Establezcamos las condiciones que deben satisfacerse en el máximo de una función de verosimilitud. Obteniendo las derivadas de $Inp(X|\pi, \mu, \mathbf{\Sigma})$ con respecto de las medias $\mu_\mathbf{k}$ e igualándolas a cero, obtenemos.

$$0 = -\sum_{n=1}^{N} \frac{\pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x_n}|\mu_\mathbf{j}, \mathbf{\Sigma_j})} \mathbf{\Sigma_k}(\mathbf{x_n} - \mu_\mathbf{k}),$$

es decir usando que:

$$\gamma(z_{nk}) = \frac{\pi_k N(\mathbf{x_n}|\mu_\mathbf{k}, \mathbf{\Sigma_k})}{\sum \pi_j N(\mathbf{x_n}|\mu_\mathbf{j}, \mathbf{\Sigma_j})},$$

obtenemos:

$$\mu_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) x_n, \tag{3.8}$$

donde hemos definido: $N_k = \sum_{n=1}^{N} \gamma(z_{nk})$. Podemos interpretar N_k como la cantidad de observaciones asignadas al cluster k.

Asi mismo si derivamos $In(X|\pi, \mu, \Sigma)$ con respecto de Σ_k y lo igualamos a cero obtenemos:

$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x_n} - \mu_\mathbf{k})(\mathbf{x_n} - \mu_\mathbf{k})^T.$$

Finalmente, maximizamos la función definida en ecuación 3.7 con respecto de π_k, teniendo en cuenta la restricción de que $\sum \pi_k = 1$. Esto lo llevamos a cabo usando multiplicadores de Lagrange y maximizando la siguiente cantidad:

$$Inp(\mathbf{X}|\pi, \mu, \mathbf{\Sigma}) + \lambda \left(\sum_{k=1}^{K} \pi_k - 1 \right)$$

Llegando a:

$$0 = \sum_{n=1}^{N} \frac{\mathcal{N}(\mathbf{x_n}|\mu_\mathbf{k}, \mathbf{\Sigma_k})}{\sum_j \pi_j \mathcal{N}(\mathbf{x_n}|\mu_\mathbf{j}, \mathbf{\Sigma_j})} + \lambda$$

Donde una vez mas aparecen las responsabilidades, multiplicando ambos lados de la ecuación por π_k y sumando sobre k haciendo uso de la restricción $\sum \pi_k = 1$. Obtenemos $\lambda = -N$. Por lo cual obtenemos:

$$\pi_k = \frac{N_k}{N} \tag{3.9}$$

Presentemos resumidamente el algoritmo de EM para mezclas Gaussianas, donde se pueden ver más detalladamente los pasos E y M, *expectation* y *maximization* respectivamente. Dada una Mezcla de distribución Gaussiana, el objetivo es maximizar la función de Verosimilitud con respecto de los parámetros $\mu_\mathbf{k}$, $\mathbf{\Sigma_k}$, π_k.

1. Inicializar las medias $\mu_\mathbf{k}$, covarianzas $\mathbf{\Sigma_k}$ y los coeficientes π_k y evaluar el valor inicial del logaritmo de la función de verosimilitud.

2. Paso E: Evaluar las responsabilidades usando los parámetros actuales.

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x_n}|\mu_\mathbf{k}, \mathbf{\Sigma_k})}{\sum_j \pi_j \mathcal{N}(\mathbf{x_n}|\mu_\mathbf{j}, \mathbf{\Sigma_j})}$$

3. Paso M: Re-estimar los parámetros usando las actuales responsabilidades.

$$\boldsymbol{\mu_k}^{nuevo} = \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})\mathbf{x_n}$$

$$\boldsymbol{\Sigma_k}^{nuevo} = \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})(\mathbf{x_n}-\boldsymbol{\mu_k})(\mathbf{x_n}-\boldsymbol{\mu_k})^T$$

$$\pi_k^{nuevo} = \frac{N_k}{N}$$

donde $N_k = \sum_{n=1}^{N}\gamma(z_{nk})$

4. Evaluar el logaritmo de la función de verosimilitud definida en ecuación 3.7 y chequear para la convergencia de los parámetros. Si la convergencia no se satisface, regresar al paso 2.

A modo explicativo veremos como se comportan los pasos *E - expectation* y *M - maximization* en la estimación de distribuciones gaussianas para datos simulados en una la recta real.

Simularemos datos de tres grupos de distribuciones normales con los siguientes parámetros, con $n = 600$ observaciones para cada agrupación y los parámetros poblacionales μ y σ dados por:

$$\mu = (-10.5, 10.5, 0) \quad \text{y} \quad \sigma = (2.5, 1.8, 3.6) \tag{3.10}$$

```
n ,mu, sigma = 600, [-10.5, 10.5, 0], [2.5, 1.8, 3.6]
x1 = np.random.normal(mu[0], np.sqrt(sigma[0]), n)
x2 = np.random.normal(mu[1], np.sqrt(sigma[1]), n)
x3 = np.random.normal(mu[2], np.sqrt(sigma[2]), n)
X = np.array(list(x1) + list(x2) + list(x3))
np.random.shuffle(X)

def pdf(data, mean: float, variance: float):
    s1 = 1/(np.sqrt(2*np.pi*variance))
    s2 = np.exp(-(np.square(data - mean)/(2*variance)))
    return s1 * s2
```

Inicializamos los parámetros a aprender del modelo, medias, varianzas, y proporciones e iteraremos en 6 pasos, mostratremos a modo ilustrativo como las distribuciones gaussianas van realmente aproximándose a las distribuciones originales figura 3.6.

En cuanto a la convergencia del modelo GMM, el modelo no necesariamente converge. A pesar de esto, se demuestra formalmente en (Zhao, Li, and Sun 2020) un estudio para la convergencia del algoritmo EM aplicado en GMM con un número arbitrario de componentes y distintos pesos. Se muestra que ha medida que los centroides de los componentes estén separados por lo menos por una cierta constante que depende de la cantidad de aglomeraciones y de la dimensión del espacio, el algoritmo EM converge localmente al óptimo global del logaritmo de la verosimilitud. También se muestra que la razón de convergencia es lineal.

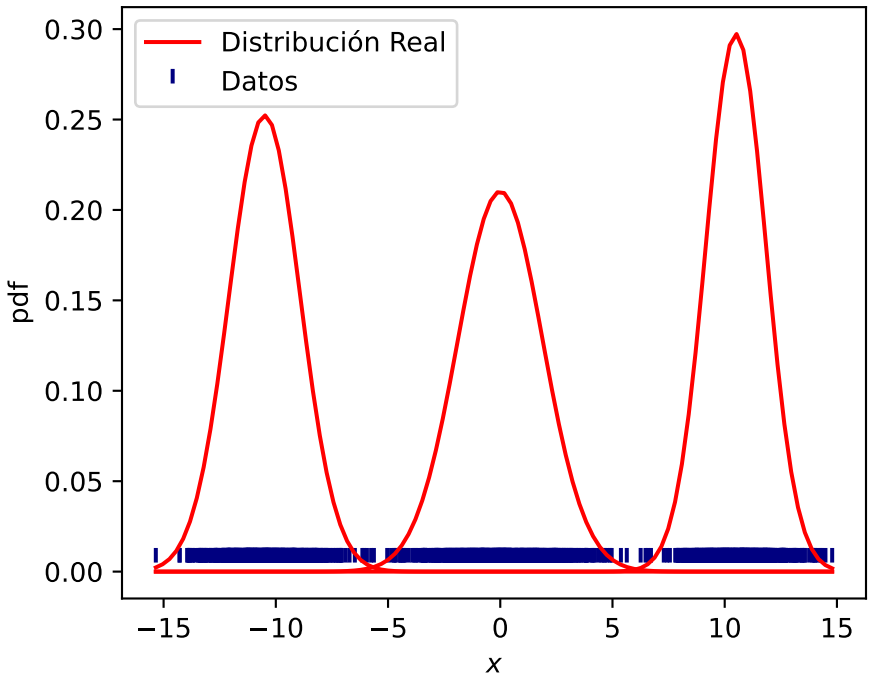

Fig 3.5: Distribución real de los datos simulados en una dimensión.

Ahora veremos como GMM soluciona algunos de los problemas que tiene K-means, esto explicándose porque GMM pretende estimar la matriz de covarianzas de cada agrupación dando asi una noción de *forma* de cada cluster, recreamos el clustering usando GMM para las figuras figura 3.2, figura 3.4, figura 3.3.

GMM logra identificar correctamente los 3 clusters, solucionando el problema de varianzas distintas, al estimar la matriz de covarianzas para cada agrupación detecta la particularidad de cada agrupación figura 3.8. Se reconocen agrupaciones muy *condensadas* y agrupaciones *dispersas*, más allá de que comparten un espacio, o en este caso incluso un cluster esta *adentro* de otro. Para datos anisotrópicos, GMM reconoce la forma de los 3 clusters, detectando la naturaleza de los datos figura 3.9 y agrupándolos adecuadamente. Aún así GMM tampoco logra solucionar esta clase de datos irregulares figura 3.10. Existen en la literatura algoritmos ampliamente conocidos que resuelven esta problemática bastente bien pero no son analizados en esta tesis, quizá el más renombrado es DBSCAN por sus siglas en inglés (Density-Based Spatial Clustering of Applications with Noise) (Schubert et al. 2017), este algoritmo tiene una complejidad temporal de $O(nlog(n))$ mientras que GMM tiene una complejidad de $O(nkd^3)$, donde k es la cantidad de aglomeraciones y d es la dimensión del espacio. Además de ser de mayor complejidad existe una gran cantidad de parámetros que el algoritmo DBSCAN necesita y la estimación de estos parámetros puede ser compleja si se debe aplicar para múltiples datasets (como es nuestro caso), la alternativa de ir retocando manualmente parámetros, o tratar de estimarlos como un problema aparte para luego usar DBSCAN, no lo consideramos una posibilidad viable.

Relación entre K-means y GMM

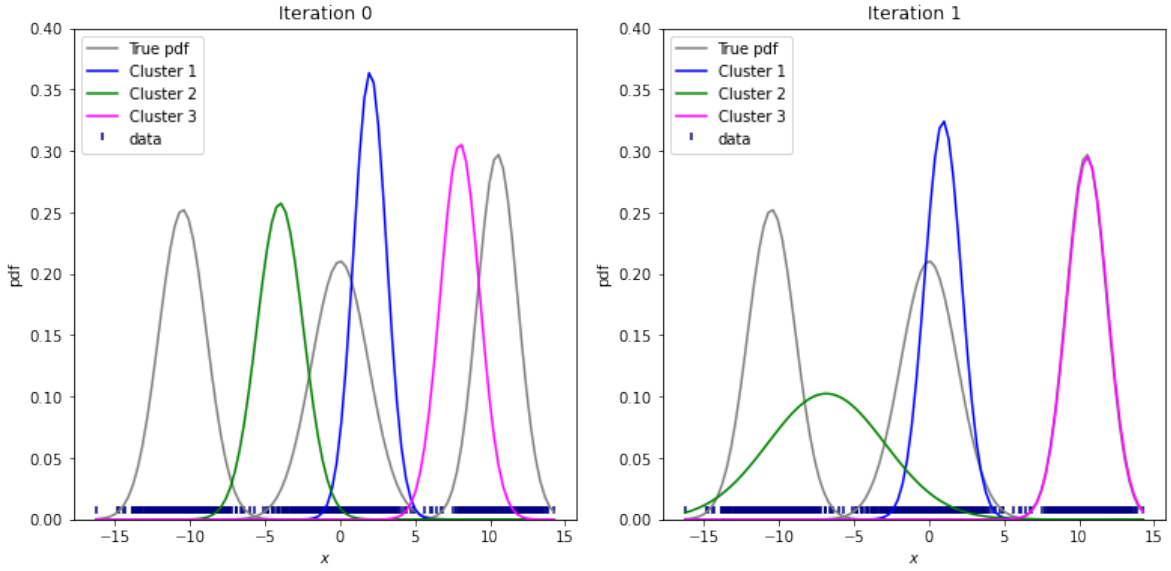

Fig 3.6: Comenzamos con las distribuciones iniciales aleatorias

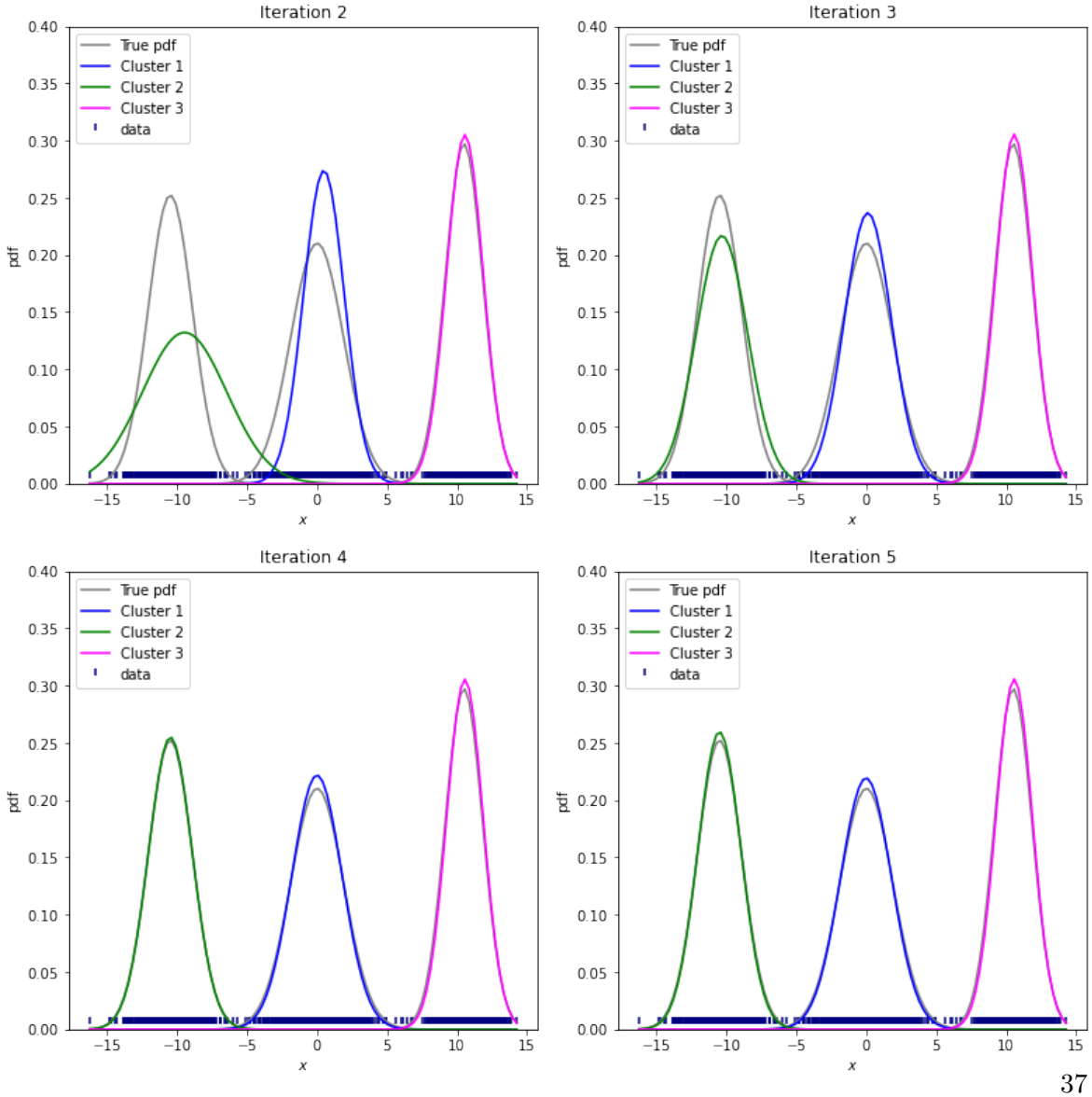

Fig 3.7: Se logra una un error aceptable.

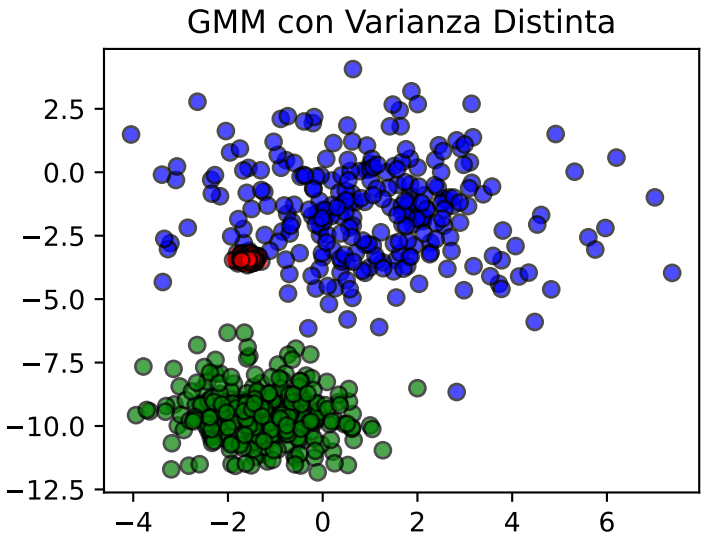

Fig 3.8: GMM para varianzas distintas.

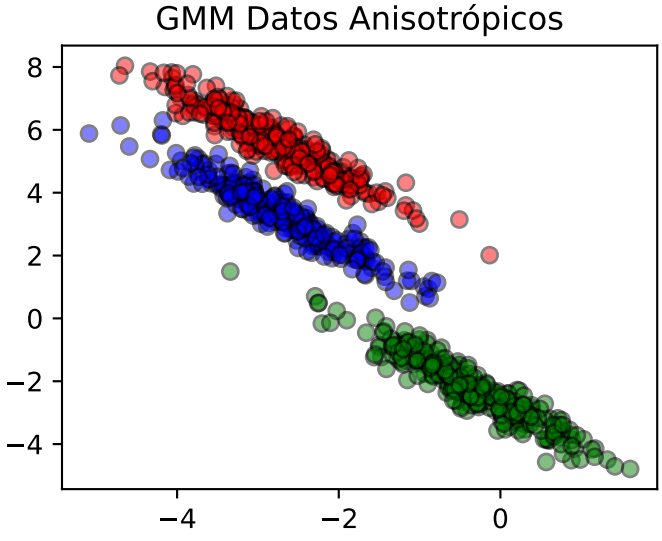

Fig 3.9: GMM para datos anisotrópicos.

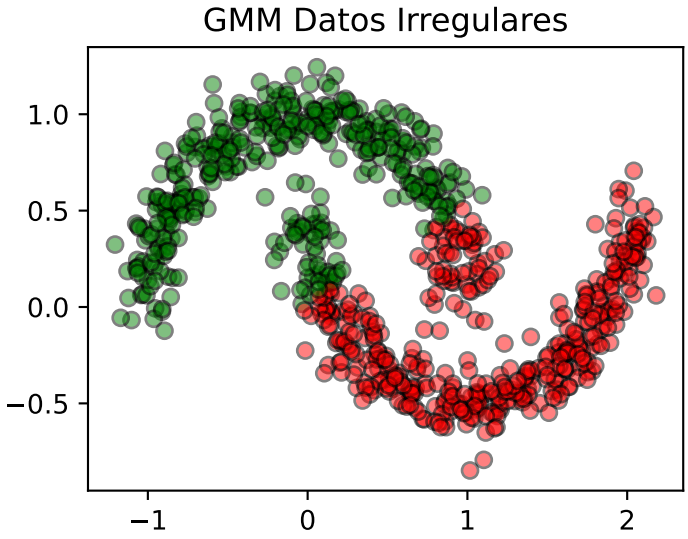

Fig 3.10: GMM Datos Irregulares

K-means se puede pensar como una particularidad de GMM pues es aplicar GMM considerando que cada grupo es una esfera uniforme, mientras que en GMM cada grupo puede tener formas esféricas arbitrarias siempre que sean esferas de distribuciones normales.

Matemáticamente no es complicado probar que si consideramos un modelo de mezclas Gaussinas en el que las matrices de covarianza de los componentes están dadas por aI (pues se consideran formas esféricas) y donde a es un parámetro compartido por todos los grupos e I es la matriz identidad, entonces obtenemos:

$$p(x|\mu_k, \Sigma_k) = \frac{1}{(2\pi a)^{1/2}} \exp\left\{-\frac{1}{2a}\|x - \mu_k\|^2\right\}$$

Ahora aplicamos el método EM para maximizar la verosimilitud en una mezcla de K distribuciones normales con a parámetro fijo (no debe ser re-estimado por el modelo). Usando la ecuación 3.6 obtenemos:

$$\gamma(z_{nk}) = \frac{\pi_k \exp\{-\|x_n - \mu_k\|^2/2a\}}{\sum_j \pi_j \exp\{-\|x_n - \mu_j\|^2/2a\}}$$

Si consideramos el limite $a \to 0$ podemos observar que en el denominador el término para el cual $\|x_n - \mu_j\|^2$ es el más pequeño va a tender a cero más lentamente y de esta manera las responsabilidades $\gamma(z_{nk})$ para la observación x_n tenderán todos a cero excepto para el término j-ésimo, en el cual $\gamma(z_{nk}) \to 1$. Observemos que esto se cumple independientemente de los valores de π_k siempre que ninguno de los π_k sea cero. Por lo tanto, en este límite, obtenemos una asignación estricta de puntos de datos a grupos, tal como en el algoritmo K-means, de modo que $\gamma(z_{nk}) \to r_{nk}$, donde r_{nk} está definido en la ecuación 3.1. Cada punto por lo tanto es asignado al cluster cuyo centroide se encuentra más proximo.

De esta manera en el paso M al re-estimar el parámetro μ_k dado por ecuación 3.8 se obtiene ecuación 3.4. Tenga en cuenta que la fórmula de re-estimación de los coeficientes de mezcla ecuación 3.9 simplemente restablece el valor de π_k para que sea igual a la fracción de puntos de datos asignados al grupo k, aunque estos parámetros ya no juegan un papel activo en el algoritmo.

Finalmente tomando el límite $a \to 0$ el logaritmo de la verosimilitud esta dado por:

$$lnp(X|\mu, \Sigma, \pi) \to -\frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}\|x_n - \mu_k\|^2 + const$$

Que como observamos es el equivalente al problema de miniminización en K-means, pues para maximizar el logaritmo de la verosimilitud se debe minimizar la función J ecuación 3.2.

Ahora veremos como los algoritmos presentados no arrojan resultados deseados para conjuntos de datos con ruido y valores atípicos. Presentaremos algoritmos que pretenden solucionar estos problemas.

3.3 Clustering Robusto

Hasta ahora hemos hecho clustering para datos que no presentan ruido. Ahora veremos como podríamos abordar el problema en el caso de que los datos estén contaminados. Existen en la literatura múltiples enfoques de como podría solucionarse este problema, estos pueden enmarcarse varias categorías principales:

1. *Enfoque paramétrico*: Se usan distribuciones con colas más pesadas, como lo puede ser por ejemplo la distribución t-student multivariada (Peel and McLachlan 2000). Partiendo de este enfoque existen múltiples mejoras en la literatura, en la que se puede agregar un componente asociado con una distribución paramétrica (posiblemente inadecuada), para capturar valores atípicos (Banfield and Raftery 1993) (Coretto and Hennig 2016) (Farcomeni and Punzo 2020). Este enfoque no será tratado en esta tesis y puede ser tratado como alternativa en un trabajo futuro.

2. *Enfoque de Poda o Trimming*: Consiste en podar un cierto porcentaje de datos que se encuentran *alejados*, de tal forma que estos posibles valores atípicos pueden llegar a tener un peso despreciable en las estimaciones (Luis A. García-Escudero et al. 2008), (Luis Angel García-Escudero et al. 2010). Este será el enfoque que se usará en este trabajo para robustecer.

3. *Enfoque de Minimización de la Escala Tau*: Desarrollado en (González 2019). Se proponen alternativas robustas: K-TAU, RMBC, basadas en minimizar una escala robusta de tipo tau de las distancias entre los puntos y los centros de los grupos a los que pertenecen. La escala Tau definida por Victor Yohai y Ruben H. Zamar (Yohai and Zamar 1988), cumple con ser una escala robusta y eficiente cuando los datos son normales. La idea de alcance general de procedimientos robustos vía la minimización de la escala tau ya fue aplicada a diversos problemas de la estadística, por ejemplo Yohai y Zamar (Yohai and Zamar 1988) la utilizaron en modelos de regresión. Esta alternativa (K-TAU y RMBC) la intentamos llevar a cabo en nuestro trabajo para un departamento particular de la

provincia de Córdoba, en el que como se comenta en el capítulo 4 existieron problemas en el uso de memoria, la implementación del paquete de R no optimiza el uso de memoria para el volumen de datos que estamos utilizando.

Se presentarán a continuación distintos casos de datos simulados en los que existen problemas de valores atípicos y veremos como se comportan los modelos vistos hasta el momento. Más adelante observaremos una heurística que puede solucionar parcialmente el problema y veremos dos métodos de intento de robustificación de Clustering, uno basado en estimadores de Mediana y otro basado en Podas o *Trimming*.

Primero simulamos una instancia en la que solo se observa un outlier presente y observemos como los algortimos presentados se comportan. Observaremos nuestro primer ejemplo que son los datos de la figura 3.11 en el que se ven 3 agrupaciones bien definidas y una observación que parece ser un outlier. Llevamos a cabo K-means con $K = 3$, $K = 4$ y GMM con $K = 3$ y $K = 4$ vemos que en el caso de K-means con ambos valores de K el algoritmo no identifica la naturaleza de los datos, pues se agruparon 2 clusters en una sola agrupación dándonos a entender que ni siquiera en este caso K-means puede ser una posibilidad ni aún considerando un cluster más y esperando que el outlier se deposite en una agrupación. Por el contrario en GMM con los $K = 3$, $K = 4$ se obtienen mejores resultados pues se identifican adecuadamente los tres clusters, en el caso de $K = 3$ se tomo el outlier como parte de la agrupación 2, y por último para este ejemplo considerando 4 clusters se tomo el outlier como el único elemento del cluster 4, dando un resultado quizá más favorable.

Mostramos como se implementa K-means con $K = 3$, en el software estadístico R-Studio.

```
kmeans_3 <- kmeans(data, centers = 3)
```

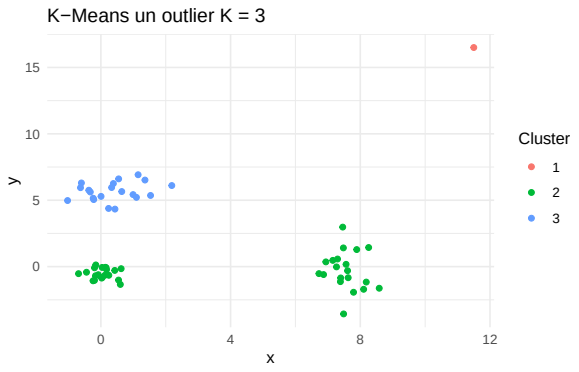

Fig 3.11: K-means, $K = 3$, con un outlier.

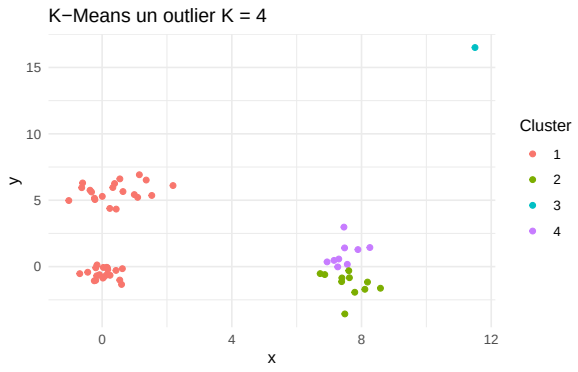

Fig 3.12: K-means, $K = 4$, con un outlier.

En ambos casos K-means no reconoce el patrón presente, GMM solucionó razonablemente el incoveniente del valor atípico usando la heurística de considerar un cluster más y de esta manera el outlier se aparte en un cluster.

Mostramos como se implementa GMM con $K = 3$, en el software estadístico R-Studio.

```
gmm_model_3 <- Mclust(data, G = 3)
```

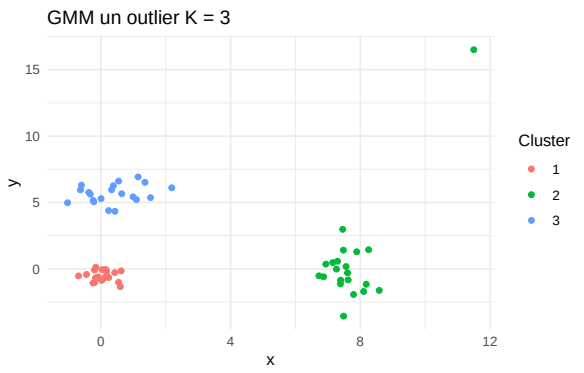

Fig 3.13: GMM $K = 3$, con un outlier.

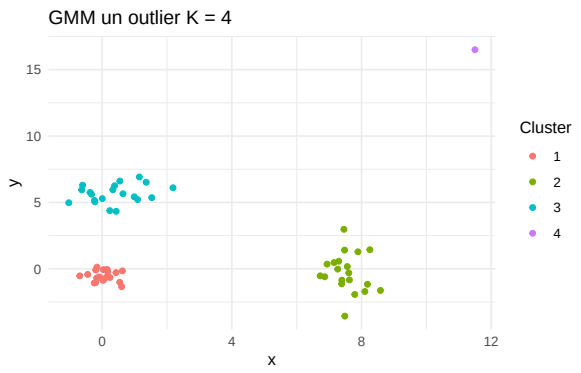

Fig 3.14: GMM $K = 4$, con un outlier.

En la figura 3.13 se detectan los clusters adecuadamente, considerando el valor atípico parte de una agrupación. En la figura 3.14 se detectan correctamente las tres agrupaciones y se considera el outlier una agrupación por si misma, lo cual es una mejora a K-means. Agregando un cluster más como se observa en la figura 3.14 se obtiene el outlier apartado en un solo cluster, lo cual es ideal.

Vemos como al inconveniente del rudio agregarle un cluster más y esperar que la agrupación adicional aparte en sí estos datos espurios puede ser considerada como una posibilidad. Quizá en algunos escenarios no sea lo más adecuado y sea necesario buscar otra estrategia, en un marco de datos más abstractos como observaciones definidas en dimensiones mayores a 3 o en escenarios en los que el ruido pueda no ser representado como un sólo dato si no como una nube de puntos problemáticos, esta heurística puede fallar. Evidentemente se presentarán algunos contextos en los que esta heurística será fallida cuando se clusterize usando GMM. Como habíamos observado en la sección sección 3.2.2, GMM es una generalización de K-means por lo que evidentemente GMM arroje mejores resultados que K-means.

3.3.1 Clustering Robusto basado en la Mediana (PAM)

Resulta natural como intento para hacer clustering robusto usar estimadores basados en la mediana por lo visto en figura 2.2. Dado que el método k-means se define a través de un criterio de mínimos cuadrados (Norma L_2), del cual hereda la falta de robustez, nos podríamos sentir tentados a tratar de robustecerlo utilizando los mismos argumentos que llevan a obtener la mediana de la muestra como una alternativa robusta, por lo visto en K-means se minimiza sobre m la expresión $\sum(x_i - m)^2$ (Norma L_2), mientras que utilizando la mediana se minimiza la expresión $\sum |x_i - m|$ (Norma L_1), por lo que se apoda en la literatura como K-medioides o k-medianas. Llegando al siguiente problema de minimización:

$$\underset{m_1,\dots m_k}{\arg\min} \sum_{i=1}^{n} \min_{j=1,\dots k} \|x_i - m_j\|$$

El enfoque L_1 dio origen al método Partitioning Around Medoids (PAM), que fue uno de los primeros intentos de realizar un análisis de conglomerados robusto. Otro enfoque L_1 también fue considerado en (Estivill-Castro and Yang 2000), en el que igualmente se proponen

medianas como estimadores de los centros. Desafortunadamente, el método PAM solo proporciona una robustez bastante tímida, se han intentado agregar a este método una cierta función de penalización pero se ha probado en la literatura que no se gana mucha robustez (Garcia-Escudero and Gordaliza 1999). El algoritmo PAM esta disponible en la libería *cluster* del software estadístico R-Studio.

Existen versiones de clustering robusto también basado en medianas que esta vez se piensa como una réplica de GMM adaptando EM (Godichon-Baggioni and Robin 2024), más específicamente el paso M pero reemplazando las estimaciones de la media y la varianza por versiones robustas basadas en la mediana y la matriz de covariación de la mediana. Todos los métodos propuestos están disponibles en el paquete R RGMM accesible en CRAN. El algoritmo es bastante reciente, publicado en el año 2024 por lo que no lo probamos a fecha de escritura de esta tesis pero se puede analizar como trabajo futuro. Este enfoque tiene algunas similitudes con el propuesto en (González 2019) pues en el trabajo citado de Juan D. Gonzáles, se modifica el algoritmo EM de modo que la estimación de parámetros sea robusta y consistente.

A continuación mostramos el resultado aplicando el clustering robusto TCLUST que formalizaremos y expondremos más adelante, se consideran para el aprendizaje $K = 3$, el modelo considera el cluster 0 como los valores *raros* y aparta estos datos como se observa en figura 3.15. Se usa la función *tclust* del software estadístico R, los parámetros de esta función serán explicados más adelante.

```
trim_clust <- tclust(data, k = 3, alpha = 0.01)
```

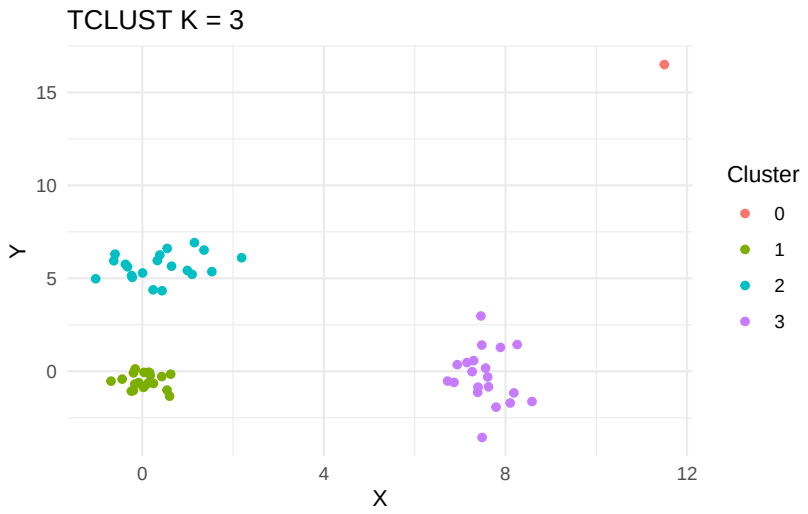

Fig 3.15: Clusters generados por TCLUST K = 3

Una vez echa una comparación sencilla con datos simulados de PAM y de TCLUST, formalizaremos el método de clustering robusto basado en Trimming y además experimentaremos con varios escenarios para analizar la capacidad de los modelos de clustering no robusto, la heurística presentada en figura 3.15 y el clustering robusto de aprender la naturaleza de los datos y lidiar con valores atípicos.

3.3.2 Método Robusto basados en Poda (Trimming)

Para la formalización matemática del problema consideraremos un conjunto de datos $x_1, .., x_n$ en $\mathbb{R}^p$. La densidad de la distribución p-variada con esperanza μ y matriz de covarianzas Σ, y k entero que denota la cantidad de grupos.

Bajo el modelo con valores atípicos, la función de varosimilitud está dada por:

$$\prod_{j=1}^{k} \prod_{i \in R_j} \pi_j f(x_i; \mu_j, \Sigma) \prod_{i \notin R} g_i(x_i) \tag{3.11}$$

Con:

$$R = \bigcup_{j=1}^{k} R_j \quad \text{y} \quad \#R = n - [n\alpha] \tag{3.12}$$

Adicionalmente para las restricciones de las matrices de covarianzas $\Sigma_j' s$ definimos una constante c, mayor o igual a 1 tal que:

$$\frac{M_n}{m_n} \leq c \tag{3.13}$$

Donde:

$$M_n = \max_{j=1,...k} \max_{l=1,...p} \lambda_l(\Sigma_j) \quad \text{y} \quad m_n = \min_{j=1,...k} \min_{l=1,...p} \lambda_l(\Sigma_j) \tag{3.14}$$

Siendo $\lambda_l(\Sigma_j)$ los autovalores de las matrices Σ_j, $l = 1, ..., p$ y $j = 1, ..., k$. Notemos que $c = 1$ produce la restricción mas fuerte posible, es decir las agrupaciones deben tener la misma forma.

Un mejor entendimiento de nuestro problema es obtenido introduciendo las funciones de asignación z_j, $j = 0, 1..., k$. Para toda observación x en $\mathbb{R}^p$, definimos $z_j(x) = 1$ si x es asignado a la clase R_j, $j = 1, ..., k$, o $z_0(x) = 1$ si el dato es recortado, a través de estas funciones. Por lo probado en (Gallegos and Ritter 2005) podemos evitar la contribución de las g_i (contribución espuria) cuando las g_i's satisfacen la siguiente condición:

$$\arg\max_{\mathcal{R}} \max_{\mu_j, \Sigma_j} \prod_{j=1}^{k} \prod_{i \in R_j} \pi_j f(x_i; \mu_j, \Sigma_j) \subseteq \arg\max_{\mathcal{R}} \prod_{i \notin \cup_{j=1}^{k} R_j} g_i(x_i) \tag{3.15}$$

Donde $\mathcal{R}$ es el conjunto de todas las posibles particiones en el conjunto de índices $\{1, \ldots, n\}$ en k grupos de observaciones regulares. Nótese que el lado derecho en la condición ecuación 3.15 sólo involucra las observaciones no regulares y no depende de la partición de las regulares. Por lo tanto, simplemente significa que cualquier conjunto de observaciones no regulares en cada partición óptima que maximice ecuación 3.11 también podría obtenerse como un subconjunto de $[n\alpha]$ elementos de la muestra que maximiza la probabilidad correspondiente al

ruido. Esta condición se cumple fácilmente bajo supuestos razonables para las g_i siempre que las observaciones no regulares puedan verse como meramente *ruido*.

Suponiendo que g_i's satisfacen ecuación 3.15 y por ende se pueden omitir, podemos plantear nuevamente el problema de maximización en ecuación 3.7.

$$\prod_{i=1}^{n}\prod_{j=1}^{k}\pi_j^{z_j(x_i)}f(x_i;\mu_j,\Sigma_j)^{z_j(x_i)}$$

Donde z_j son funciones binarias tales que $\sum_{j=0}^{k} z_j(x_i) = 1$ y $\sum_{i=1}^{n} z_0(x_i) = [n\alpha]$. Tomando logaritmos llegamos a la siguiente formulación de maximización bajo las restricciones dadas en ecuación 3.13 y ecuación 3.14:

$$\mathbb{E}\left(\sum_{j=1}^{k} z_j(.)(Ln\pi_j + Lnf(.;\mu_j,\Sigma_j))\right)$$

En términos de las funciones de asignación:

$z_j : \mathbb{R}^p \to \{0,1\}$ tal que $\sum_{j=0}^{k} z_j = 1$ y $Ez_0(.) = \alpha$

Y los parámetros $\theta = (\pi_1,...,\pi_k,\mu_1,...,\mu_k,\Sigma_1,...,\Sigma_k)$ con los pesos $\pi_j \in [0,1]$, con $\sum_{j=1}^{k}\pi_j = 1$, $\mu_j \in \mathbb{R}^p$ y matrices definidas positiva y simétricas Σ_j, $j = 1,...,k$.

Ahora pasaremos a simplificar notablemente nuestro problema por medio de una adecuada reformulación que nos lleva a expresar las funciones de asignación $z_j's$ sólo en términos de θ. Dado $\theta \in \Theta$, consideramos las *funciones discriminantes* como:

$$D_j(x;\theta) = \pi_j f(x;\mu_j,\Sigma_j) \quad \text{y} \quad D(x;\theta) = \max\{D_1(x;\theta),\dots D_k(x;\theta)\} \tag{3.16}$$

Estas funciones pretenden medir que tan *atípica* es una observación.

Utilizando definiciones anteriores ecuación 3.16, para un θ dado y una medida de probabilidad P, consideramos la función de distribución de $D(\,\cdot\,;\theta)$ y su α-cuantil correspondiente:

$$G(u;\theta,P) := P(D(\,\cdot\,;\theta) \le u) \quad \text{y} \quad R(\theta,P) := \inf_u G(u;\theta,P) \ge \alpha.$$

Con esta notación y escritura podemos caracterizar las funciones de asignación y llegar a la siguiente simplificación del problema de clustering robusto.

Buscamos maximizar sobre θ la función $L(\theta,P)$. que definimos como:

$$L(\theta,P) := E_P\left[\sum_{j=1}^{k} z_j(\,\cdot\,;\theta)\log D_j(\,\cdot\,,\theta)\right] \tag{3.17}$$

Donde las funciones de asignación se obtienen a partir de θ como:

$$z_j(x;\theta) = I\{x : \{D(x;\theta) = D_j(x;\theta)\} \cap \{D_j(x;\theta) \geq R(\theta, P)\}\} \quad \text{y} \quad z_0(x;\theta) = 1 - \sum_{j=1}^{k} z_j(x;\theta)$$

Es decir, asignamos x a la clase j con el mayor valor de función discriminante $D_j(x;\theta)$ o x se elimina cuando todos los $D_j(x;\theta)$ (y, en consecuencia, $D(x;\theta)$) son menores que $R(\theta, P)$. (Para desempatar en los valores de la función discriminante, se podría aplicar el orden lexicográfico). Los resultados matemáticos relevantes a considerar están probados en (Luis A. García-Escudero et al. 2008).

El Algoritmo TCLUST se puede ver como:

1. Seleccionar aleatoriamente los centroides m_j^0's, las matrices de covarianza S_j^0's y los pesos de los grupos p_j^0's para $j = 1, ..., k$.

2. De los parámetros $\theta^l = (p_1^l, .., p_k^l, m_1^l, .., m_k^l, S_1^l, .., S_k^l)$ obtenidos anteriormente.

   - (**Poda**) Obtener las distancias $d_i = D(x_i, \theta^l)$ para las observaciones $x_1, ..., x_n$ y mantener el conjunto H con $[n(1-\alpha)]$ observaciones.
   - Separar H en $H_1, ..., H_k$ con $H_j = \{x_i \in H : D_j(x_i, \theta^l) = D(x_i, \theta^l)\}$.
   - Obtener el número de puntos n_j en H_j y su media muestral y covarianza muestral, m_j y S_j, $j = 1, .., k$.
   - Considerar la descomposición en valores singulares de $S_j = U_j' D_j U_j$ donde U_j es una matriz ortogonal y D_j la matriz diagonal de autovalores. Si el vector de autovalores no satisface la condiciones ecuación 3.13 y ecuación 3.14, obtenemos usando por ejemplo el Algoritmo de Dykstra's un nuevo vector que si satisfaga esa condición.
   - Actualizar θ^{l+1} usando:
     - $p_j^{l+1} \leftarrow n_j/[n(1-\alpha)]$
     - $m_j^{l+1} \leftarrow m_j$
     - $S_j^{l+1} \leftarrow U_j' \hat{D}_j U_j$ y $\hat{D}_j = diag(\Lambda_j)^{-1}$.

3. Realizar F iteraciones del proceso descrito en el paso 2 (valores moderados para F suelen ser suficientes) y calcule la función de evaluación $L(\theta^F; P_n)$ utilizando la expresión ecuación 3.17.

4. Comenzar desde el paso 1 varias veces, manteniendo las soluciones que conducen a valores mínimos de $L(\theta^F, P_n)$ para elegir la mejor.

Este algoritmo nos lleva a una asignación óptima, por lo probado en (Luis A. García-Escudero et al. 2008). Como se explicó anteriormente el término α representa la proporción de poda.

En TCLUST, como se mencionó busca eliminar observaciones que a juicio del modelo sean atípicos, esto lo llevamos a cabo esencialmente con el concepto de *trimm* (en inglés), escencialmente *podamos* los datos, analizamos un subconjunto de los datos originales. Una vez formalizado el algoritmo TCLUST, comparémoslo con PAM para datos simulados figura 3.16. Vemos un mejor comportamiento del algoritmo TCLUST para este panorama sencillo de datos simulados.

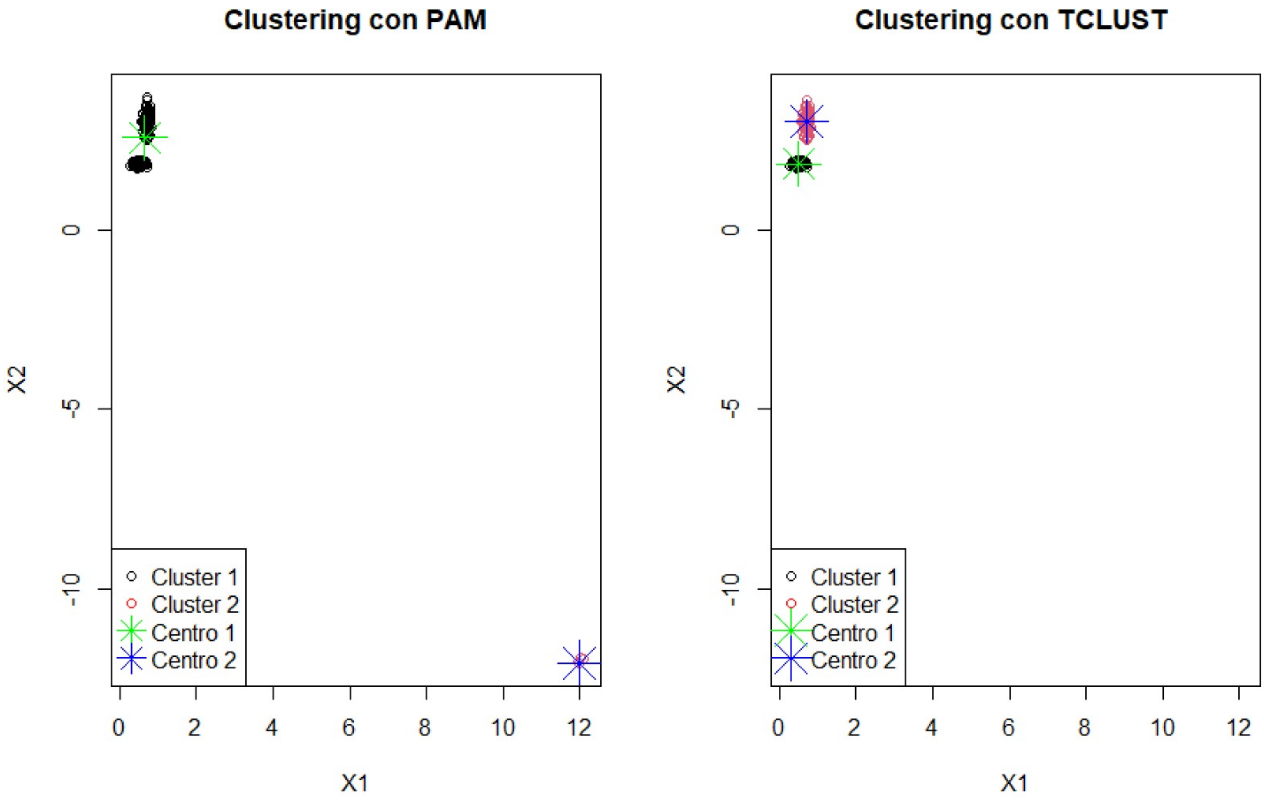

Fig 3.16: Comparación entre PAM y TCLUST, vemos como PAM no puede solucionar este problema con pocos valores atípicos. TCLUST funciona correctamente.

Más simulaciones. K-Means, GMM, TCLUST

Veremos como se comporta K-means, GMM y el método TCLUST. Las simulaciones son llevadas a cabo usando R-Studio, se agrega en algunos casos para K-means y GMM un cluster más, para ver en que instancias aplicar esta heurística, puede apartar el ruido en un grupo.

Outliers Dispersos

A modo de observación simulamos unos datos que tienen 2 aglomeraciones marcadas figura 3.17 y una nube de puntos que parecen representar ruido y mediciones fallidas.

Aplicando K-means $K = 3$ intentando que K-means detecte esos valores corruptos nos encontramos con un mal comportamiento figura 3.18. Evidentemente el algoritmo para $K = 2$ figura 3.17 y $K = 3$ no presiente outliers, y considera todos los datos *confiables*, en ambos casos K-means no arroja resultados satisfactorios. Ahora haciendo GMM con $K = 2$ figura 3.19

se observa que como esperamos GMM si detecta un cluster efectivamente, la segunda agrupación la toma como con el ruido incluido. Para GMM $K = 3$ figura 3.20 se toma el ruido como una sola agrupación, y los otros dos clusters se reconocen apropiadamente, con lo cual para este caso la heurística de tomar un cluster más esperando que este reconozca el ruido si funcionó.

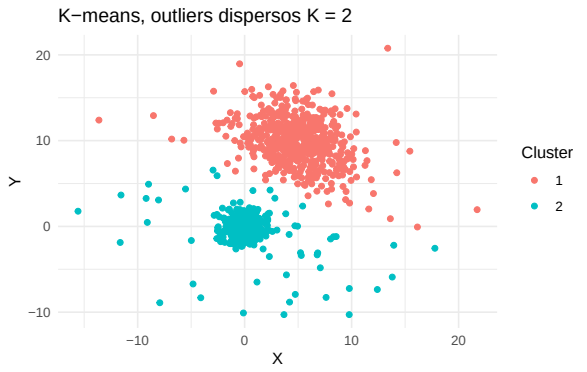

Fig 3.17: K-means K=2. Para datos con outliers dispersos.

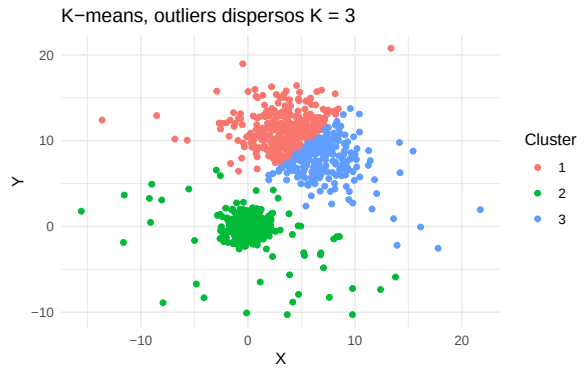

Fig 3.18: K-means K=3. Para datos con outliers dispersos.

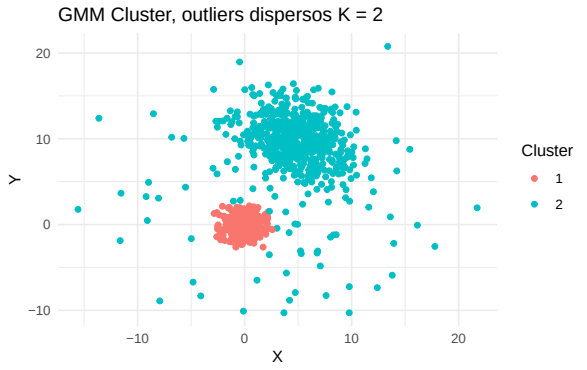

Fig 3.19: GMM K=2. Para datos con outliers dispersos.

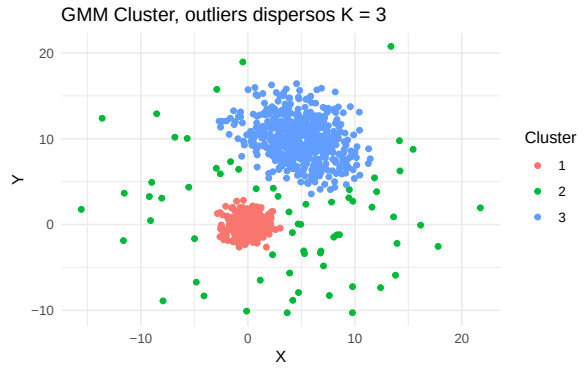

Fig 3.20: GMM K=3. Para datos con outliers dispersos.

Vemos que el cluster 0 es el que usa el método TCLUST para agrupar los outliers figura 3.21, los demás cluster corresponden a las agrupaciones correctamente identificadas. En la libreria de R llamada *tclust* se asocia al cero con los valores atípicos figura 3.29, como es explicado en (Fritz, Garcia-Escudero, and Mayo-Iscar 2012).

Subespacios con ruido

Veremos como se comporta el modelo para subespacios de una determinada forma, en un ejemplo muy interesante donde es de interés ver si se reconocen los patrones que subyacen en los datos. K-means con 3 clusters no detecta ningún cluster, figura 3.22. Si se detectó un cluster correctamente usando GMM como se ve en figura 3.23, con un cluster más $K = 4$ usando GMM no se reconocen satisfactoriamente los grupos como vemos en la figura 3.24. Si se logró usando clustering robusto TCLUST, detectar el ruido representado por el cluster rojo y las tres otras agrupaciones, observando la figura 3.25. En la libreria de R llamada *tclust*

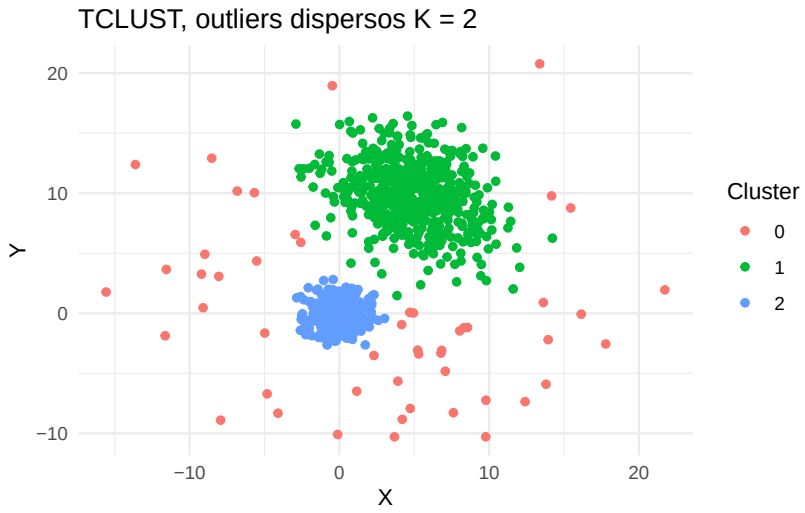

Fig 3.21: TCLUST K=2, para datos con outliers dispersos.

se asocia al cero con los valores atípicos, como es explicado en (Fritz, Garcia-Escudero, and Mayo-Iscar 2012).

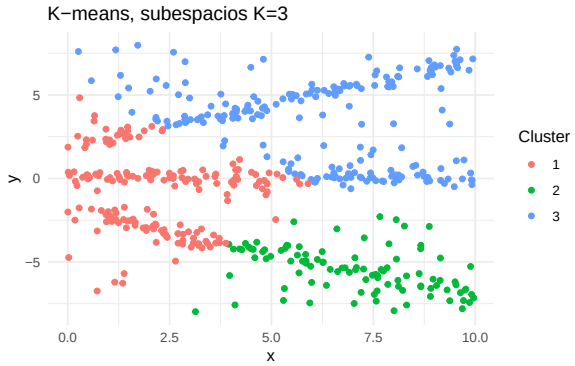

Fig 3.22: K-means para subespacios K=3.

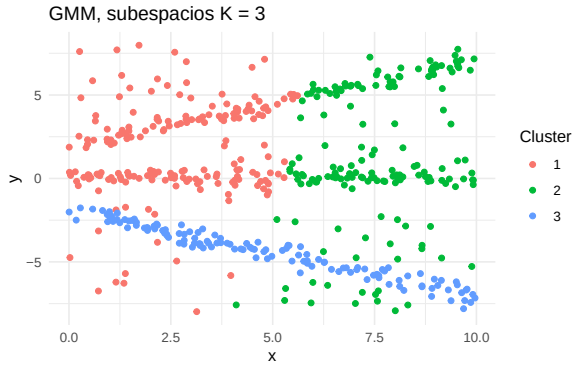

Fig 3.23: GMM para subespacios K=3.

Simulación M5data

El conjunto de datos que vemos en la figura 3.26 son datos generados por el conjunto de datos *M5data* del software estadístico R-Studio. Se observan 3 agrupaciones marcadas y un ruido en todos los datos. Cuando hacemos K-means con $K = 3$ se identifica sólo un grupo correctamente figura 3.26, usando GMM $K = 3$ se identifican satisfactoriamente 3 grupos, pero no *reconoce* el ruido presente y lo agrupa dentro del cluster mas razonable figura 3.27. Usando GMM con 4 grupos el modelo no logra agrupar el ruido presente en un cluster figura 3.28, pues en el marco del problema el modelo no se dedica a *buscar* valores atípicos. Con TCLUST vemos que efectivamente si aprende la matriz de covarianzas y se reconocen los outliers. En la libreria de R llamada *tclust* se asocia al cero con los valores atípicos figura 3.29, como es explicado en (Fritz, Garcia-Escudero, and Mayo-Iscar 2012).

Por lo visto en los ejemplos y panoramas presentados concluimos que el método TCLUST logra aprender los centros y la forma de las agrupaciones estimando las matrices de covarianzas

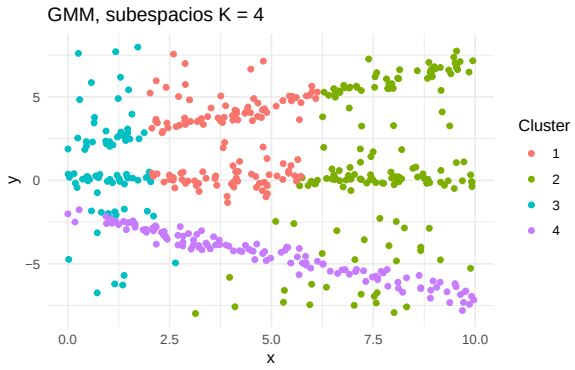

Fig 3.24: GMM para subespacios K=4.

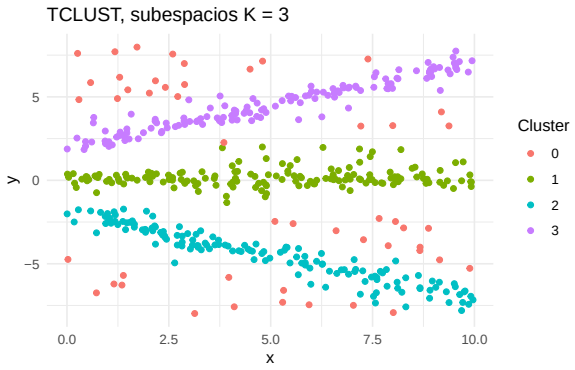

Fig 3.25: TCLUST para subespacios K=3.

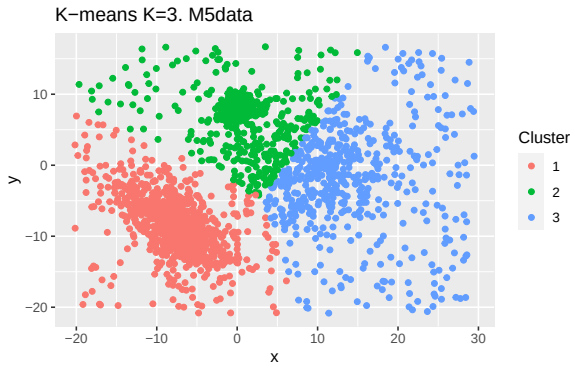

Fig 3.26: K-means K=3.

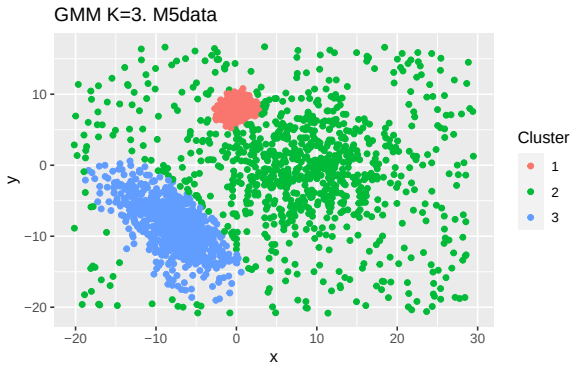

Fig 3.27: GMM K=3.

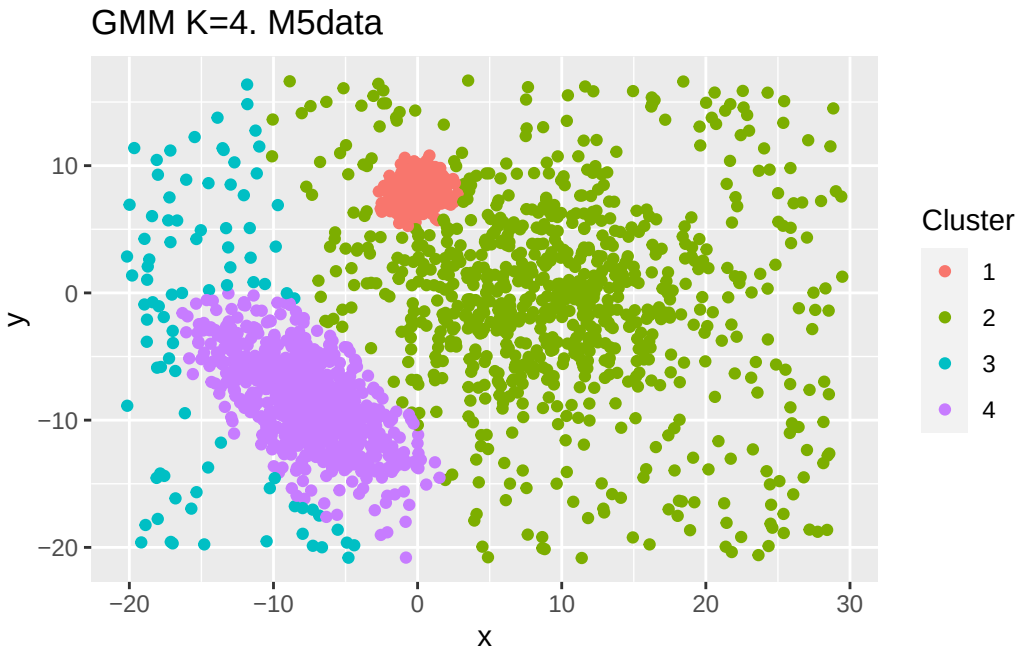

Fig 3.28: GMM K=4 para datos con ruido

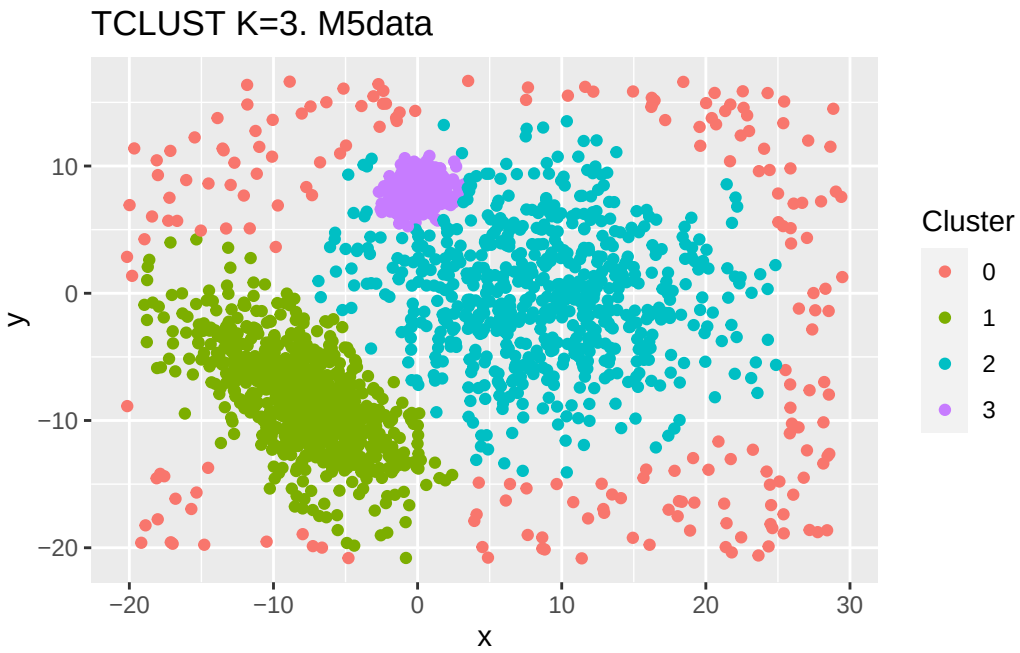

Fig 3.29: TCLUST K=3 para datos con ruido

de los grupos, además de reconocer los datos atípicos robustesiendo el algoritmo de clustering. Se presencia la necesidad con datos simulados de emplear alternativas robustas para obtener resultados más precisos y se evidencia como no emplear clustering robusto puede sesgar nocivamente los resultados. Ahora en más y habiendo justificado con varias simulaciones de la necesidad y capacidad del algoritmo TCLUST optaremos por usar este algoritmo en los datos reales.

4 Aplicación a Datos Reales

4.1 Filtro Hampel y tratamiento de Series de Tiempo.

Existen datos faltantes y outliers en cada serie de tiempo, esto requiere un tratamiento para limpiar estos datos, *pulirlos* de tal forma de crear objetos que almacenen en si la esencia de cada serie de tiempo así como también sean sencillos de manejar para hacer clustering, este trabajo implica probablemente el proceso más tedioso del trabajo total, pues se pretende reducir moralmente la dimensionalidad del problema obteniendo series de tiempo las cuales podamos analizar fácilmente y después de esto a cada serie de tiempo aplicarle un *feature extraction*, para solucionar el problema de datos faltantes y facilitar el aprendizaje no supervizado. Con el objetivo de comprimir se comprimieron, almacenaron y se leyeron archivos con extensión *.gzip*. Un archivo con extensión ".gzip" es un archivo comprimido usando el algoritmo de compresión Gzip (GNU zip). Gzip es un formato y un software de compresión que se utiliza comúnmente en sistemas Unix y Linux para reducir el tamaño de los archivos, en nuestro caso el sistema operativo usado fue Windows, existe en la librería *pandas* del software Python usada para el manejo de bases de datos en Python, la forma de especificar que los archivos son de esta extensión para su lectura. No estudiaremos como funciona Gzip pero si destacamos que hemos usado esta compresión en los archivos analizados y que la reducción del peso en memoria es sustancial.

Se tienen un aproximado de 349.200.000 de series de tiempo considerando los 291 departamentos agrícolas y los 6 años distintos que analizamos. Se dispone de una cantidad abrumadora de datos, por lo que por primera manipulación se eliminan manualmente las filas (es decir las series de tiempo) con una cantidad de valores no disponibles o *missing values* mayor al 80%, después por cada fila de cada departamento-año filtramos y eliminamos valores atípicos usando el método hampel sección 2.1.1, este filtro como se explicó pretende eliminar la mayor cantidad de los outliers del NDVI y reemplazan los valores atípicos detectados por un valor no disponible. Se intentó hacer imputación mediante regresión lineal para lidiar con valores faltantes en las series de tiempo, pero por razones de alta eficiencia temporal (siempre considerando la gigantesca cantidad de series de tiempo a tratar), la no reducción de la dimensionalidad y el riesgo de generar imputaciones defectuosas dada posibilidad de que se *escapen* valores atípicos que el filtro Hampel no pueda detectar no terminamos optando por esta alternativa.

A continuación presentamos dos series de tiempo que representan la problemática de tener observaciones con pocos datos (enorme cantidad de valores atípicos) figura 4.1 e instancias ruidosas figura 4.2.

Por distintas pruebas realizadas con el filtro Hampel para series de tiempo de datos reales los parámetros escogidos como tamaño de ventana y umbral son 3 y 2.5 respectivamente, no se puede garantizar que la efectividad del filtro Hampel sea perfecta dada la excesiva cantidad de series de tiempo, pero su uso se hace imperativo para limpiar el ruido y generar series de

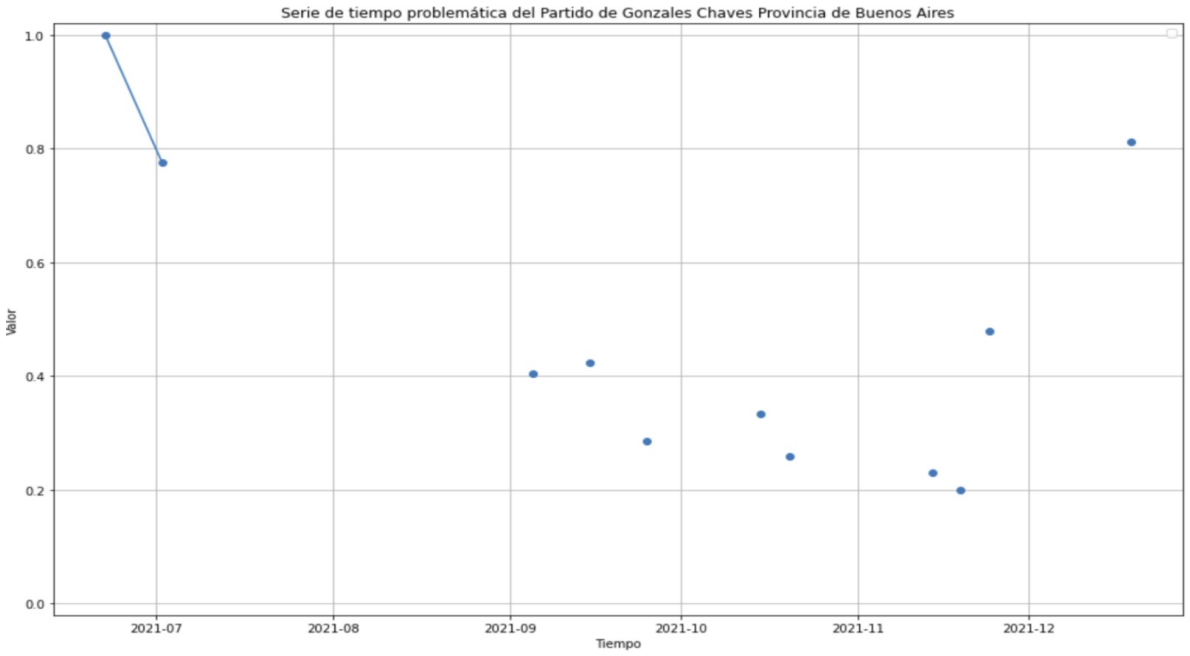

Fig 4.1: Serie de tiempo con muchos valores faltantes, partido de Gonzáles Chaves de la Provincia de Buenos Aires.

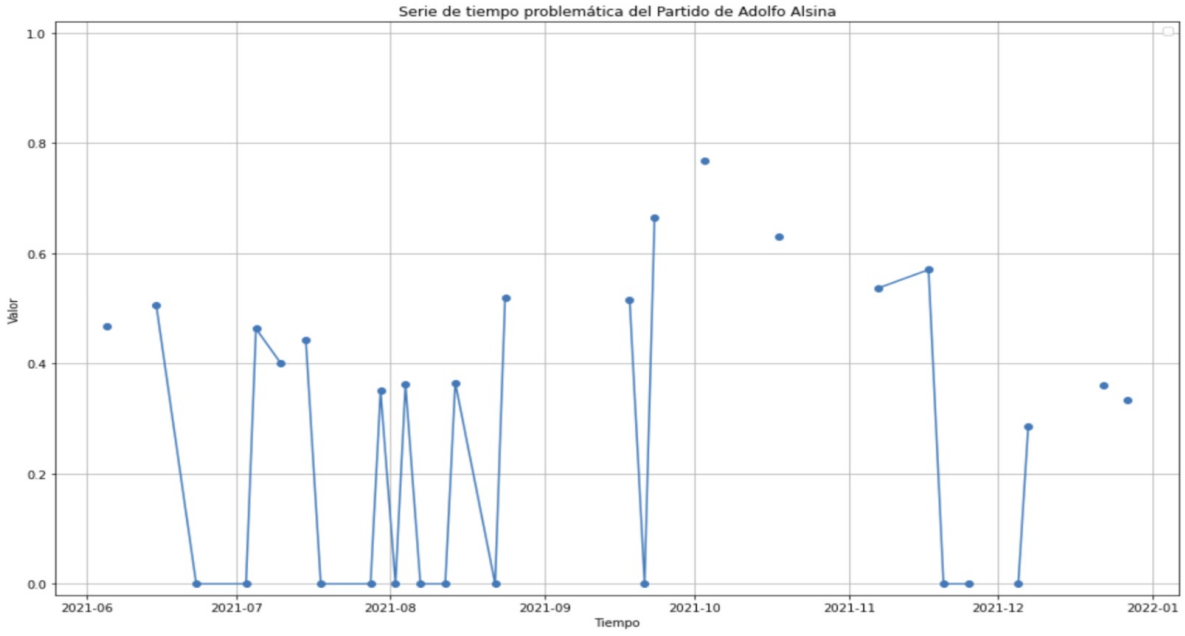

Fig 4.2: Serie de tiempo con ruido considerable, partido de Adolfo Alsina de la Provincia de Buenos Aires.

tiempo más confiables. Las series de tiempo filtradas forman otro dataset por departamento-año, estas nuevos datasets se conforman con series de tiempo que al ser filtradas usando Hampel pueden tener una cantidad de valores faltantes mayor al 80% (pues se reemplazó el valor atípico por un valor faltante), por lo que una vez más desechamos manualmente series de tiempo con un porcentaje de valores no disponibles mayor al 80% obteniendo de esta forma series de tiempo filtradas con Hampel y con filas en las que tengamos por lo menos un porcentaje mayor al 20% de valores numéricos del índice NDVI con series de tiempo realmente representativas para un posterior proceso de aprendizaje.

Notemos que se han eliminado filas antes y después de aplicar el filtro Hampel pues estas pueden contener en ambos casos pocos valores del índice NDVI y al intentar aplicar feature extraction y hacer clustering los modelos pueden sufrir innecesariamente, además las series de tiempo desechadas son poco representativas, obteniendo así por cada departamento-año *datasets* mucho más livianos en términos de memoria, haciendo de esta forma una primera reducción de la complejidad espacio-temporal del problema y llegando a series de tiempo más convenientes para extraer conclusiones.

Una vez obtenidas las series de tiempo *aceptables* se intentará solucionar el problema de los datos faltantes y además se reducirá la dimensionalidad de cada base de datos asociando cada fila es decir, cada serie de tiempo, con una función de fiteo cuyos parámetros almacenaremos en tuplas de 8 números reales, estas tuplas definen una función de una sola variable que fitea la serie de tiempo, todo esto siempre con el propósito de que el clustering que se realiza al final se lleve a cabo de una manera mucho mas eficiente y se propague el menor error posible. Obteniéndose de esta manera por departamento-año su base de datos representantiva de parámetros, existen departamentos para los cuales no se requirió llevar a cabo la extracción de parámetros pues para estos la cantidad de filas era bastante insignificante para considerar hacer clustering, la mayoría de estos datasets corresponden al año 2017 por ejemplo el partido de Villa Gesell de la provincia de Buenos Aires, el departamento de General Donovan de la provincia del Chaco, el departamento Catriló de la provincia del Chaco, entre otros. Para el año 2021 el partido al que no se aplicó extracción de parámetros fue el partido de la Costa de la Provincia de Buenos Aires.

Se llevan a cabo GMM (clustering no robusto) y TCLUST (clustering Robusto). Para cada departamento y cada año se obtienen los centroides con $K = 6$, cada centroide es el representante mas natural de cada aglomeración y de estos centroides detectamos cual de ellos es el trigo, según la información de campo.

Se clasifica el trigo en los centroides creados con las series de tiempo que alcanzan el pico de indice NDVI dentro de los meses de finales de Agosto, Septiembre, Ocutubre, y comienzos de Noviembre, los mínimos deben estar en los meses de Junio, julio y finales de la primavera. Según los datos historicos del comportamiento de la siembra, cosecha y cultivo del trigo se planta la semilla en los meses de junio y julio y se alcanza el pico a finales de invierno y durante la primavera (Molfese and Fritz 2020) . Por conocimientos de campo se establece una cantidad fija de agrupaciones $K = 6$ para todos los departamentos finalmente clusterizados.

La metodología a seguir es la siguiente:

1. *Recopilación de Datos*: Extracción de los datos satelitáles por año y por departamento de la Pampa Argentina con el uso de la Nube AWS, así como la obtención de los datos históricos de superficies de trigo cultivadas por departamento-año.

2. *Limpieza de Datos*: Filtración de valores atípicos en series de tiempo usando el filtro Hampel y el descarte de series de tiempo con una cantidad de valores faltantes mayor al 80%, tanto antes como después de aplicar el filtro Hampel.

3. *Feature Extraction*: Extracción de parámetros con el fiteo Splines-TRF, para cada departamento-año se realiza el fiteo Splines-TRF. Es decir, para cada observación (serie de tiempo) se obtienen los 8 parámetros por Splines-TRF que representan la serie de tiempo.

4. *Clustering de Series de Tiempo* $k = 6$: Clusterización tanto robusta como no robusta. Para cada departamento y cada año se obtienen los centroides de cada cluster, es decir el representante de la serie de tiempo asociada a cada aglomeración.

5. *Clasificación del Trigo*: Cada centroide obtenido del clustering es una curva representada por una tupla de 8 parámetros, se tomarán las curvas que cumplan con el requisito de *ser trigo*, esto es que tengan picos mayores a un índice de 0.6 en los meses de agosto hasta finales de octubre y además tengan valores menores a 0.4 en los otros meses. Se almacena la agrupación de esas curvas para después consultar la proporción. En el caso de no existir una curva que cumpla con los requisitos de *ser trigo* la proporción estimada de Trigo cultivada en ese departamento-año será nula.

6. *Estimación de Proporciones*: Una vez obtenida la agrupación correspondiente al trigo (en caso de no exitir la proporción estimada será nula) se calculará la proporción de observaciones que pertenecen a ese cluster, haciendo el promedio aritmético entre la cantidad de series de tiempo que están en ese cluster dividido sobre la cantidad de observaciones totales.

El algoritmo de clustering robusto aplicado en esta tesis es el TCLUST pues es el que funcionó mejor de datos simulados entre los algoritmos de clustering robusto revisados (PAM y TCLUST) y además porque la eliminación de observaciones en el proceso de aprendizaje (Trimming) es una gran ventaja en cuanto a la eficiencia y purificación obteniendo así las series de tiempo realmente representativas, que son los centroides de cada aglomeración, la explicación del funcionamiento de la función *tclust* del software estadístico R-Studio, se encuentra disponible en (Fritz, Garcia-Escudero, and Mayo-Iscar 2012) . Escoger un adecuado valor de podado α es un proceso de prueba y error y en nuestro caso es un problema de profunda dificultad pues se lleva a cabo esta clusterización para más de doscientas bases de datos y se hace necesario fijar un valor único de podado para todos los departamentos, evidentemente no se tiene control de cual es el parámetro de podado más adecuado para cada departamento particular, porque no se tiene noción del ruido en cada departamento por la complejidad misma del problema., existen esfuerzos en la literatura donde se intenta determinar el valor más apropiado de valor de podado (Fritz, Garcia-Escudero, and Mayo-Iscar 2012).

Existe una importante reducción de la dimensionalidad en cada paso a modo de ilustración se muestra el siguiente diagrama figura 4.3. En cada paso del proceso se ha llevado a cabo una importante reducción de memoria y de los datos, reduciendo así el tiempo de ejecución.

Por último es de importancia mencionar que no se tienen conocimientos de campo para varios departamentos, es decir no se disponen datos reales de la superficie cultivada de trigo para algunos departamentos del país, y para estos no fue posible establecer una comparación entre lo que predijo el modelo y la proporción real. Se pudo por lo tanto comparar 220 departamentos del país, lo cual sigue siendo una cantidad de superficie bastante considerable.

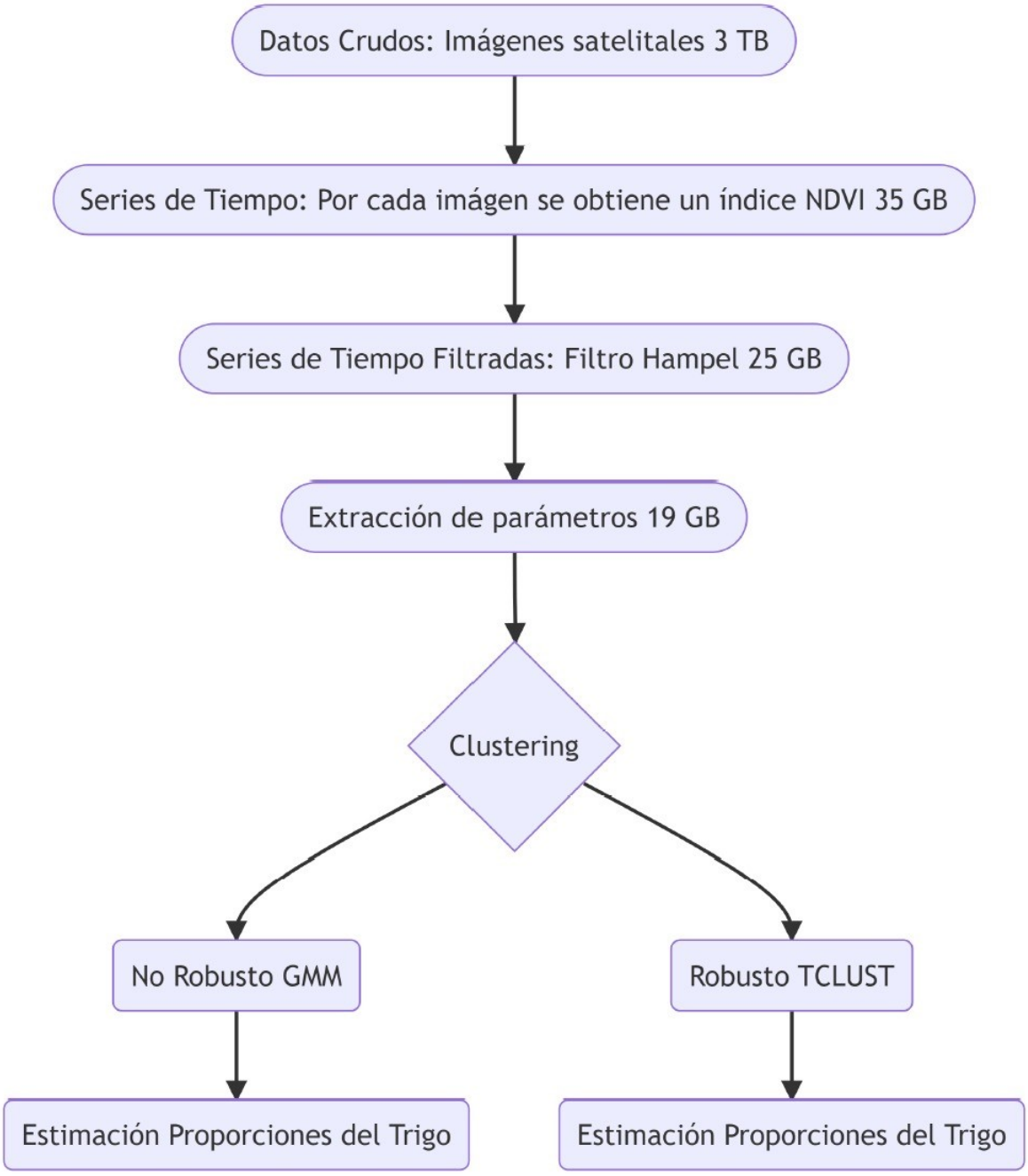

Fig 4.3: Diagrama del Procedimiento general, junto con la reducción de la memoria que se lleva a cabo en el proceso, 1TB = 1000GB.

4.2 Fiteos y Extracción de Parámetros.

Conociendo los métodos de fiteo presentados en esta tesis para hacer feature extraction de la considerable cantidad de series de tiempo, mostraremos en esta sección el comportamiento de los métodos en algunas series de tiempo de datos reales y se mostrará cual ha sido escogido, dependiendo de su capacidad explicativa considerando de primordial importancia la robustez asi como también la complejidad de los métodos. Daremos gráficos de series de tiempo con sus respectivos fiteos, usando Smoothing Splines y Splines-TRF.

Es importante recordar que el fiteo Smooth Splines precisa un parámetro de suavizado s, que como se explicó en sección 2.2.3 encontrar el más adecuado es un proceso de prueba y error, un "buen" valor de s es esperable que se encuentre entre $m - \sqrt{2m}$ y $m + \sqrt{2m}$, donde m es el número de datos. Si todos los pesos son iguales a uno, una opción razonable será estimar $s \sim m\hat{\sigma}^2$, donde $\hat{\sigma}$ es un estimado de la desviación estándar de los datos. En nuestro caso escogeremos $\hat{\sigma} = 1.4826MAD$. La desviación estándar robusta definida en el capitulo 2 sección 2.2.3. Establecer un *correcto* valor de s desmotiva a seguir la alternativa de Smooth Splines pues es necesario dada la complejidad del problema mantener este parámetro uniforme en las millones series temporales que se presentan, si bien en algunos casos se pueden generar fiteos aceptables, en otros casos se pueden tener representaciones poco confiables, y este riesgo no estamos dispuestos a asumirlo. Notar que hemos usado como parámetro s en Smoothing splines la estimación recomendada $s \sim mMAD^2$.

Observaremos en figura 4.4 una mejor adaptación del fiteo Splines-TRF asi como la ingente necesidad de aplicar el filtro hampel pues como se observa como la existencia de dos valores atípicos sesga los fiteos en ambos métodos, el escenario ideal es en el que se observa un filtrado previo (serie de tiempo filtrada con Hampel) y usando el fiteo Splines-TRF, optar por la otra alternativa puede propagar un error nocivo para el análisis cluster y de esta manera crear predicciones del trigo bastante erradas. Se observan los dos fiteos, aplicados a series de tiempo con datos reales y datos filtrados con el Método Hampel, y se ve una mejor capacidad de Splines-TRF para descibir la serie de tiempo. Asi como una mejor tolerancia a los valores atípicos que puedan llegar a presentarse, figura 4.4.

Para datos previamente filtrados con hampel analizaremos la figura 4.5 para una serie de tiempo del departamento Adolfo Alsina de la provincia de Buenos Aires año 2021 como se comportan los fiteos para Splines-TRF y Smooth Splines con dos distintos valores de parámetro de suavizado s, es importante remarcar que no se logró detectar un valor que parece ser atípico en la serie de tiempo y para este panomara un fiteo robusto es lo más ideal, vemos que Splines-TRF es efectivamente mejor.

Observamos la mayor capacidad del Método Splines-TRF para describir la serie de tiempo y también para lidiar con el ruido perjudicial (color verde), el fiteo Smooth Splines no genera resultados muy convincentes tanto para un parámetro de suavizado muy chico $s = 0$, como para el recomendado en la sección 2.2.3 definido como $s = m\hat{\sigma}^2$, para un parámetro de suavizado igual a cero se obtiene un fiteo que ni siquiera cumple con las condiciones de borde (pues $0 \leq NDVI \leq 1$) en color rojo, y para el parámetro de suavizado recomendado no se describe adecuadamente la serie de tiempo (en color azul). El índice NDVI lo analizamos entre cero y uno pues para valores negativos tenemos nubes y agua como se mencionó en el la introducción de este trabajo sección 1.1.

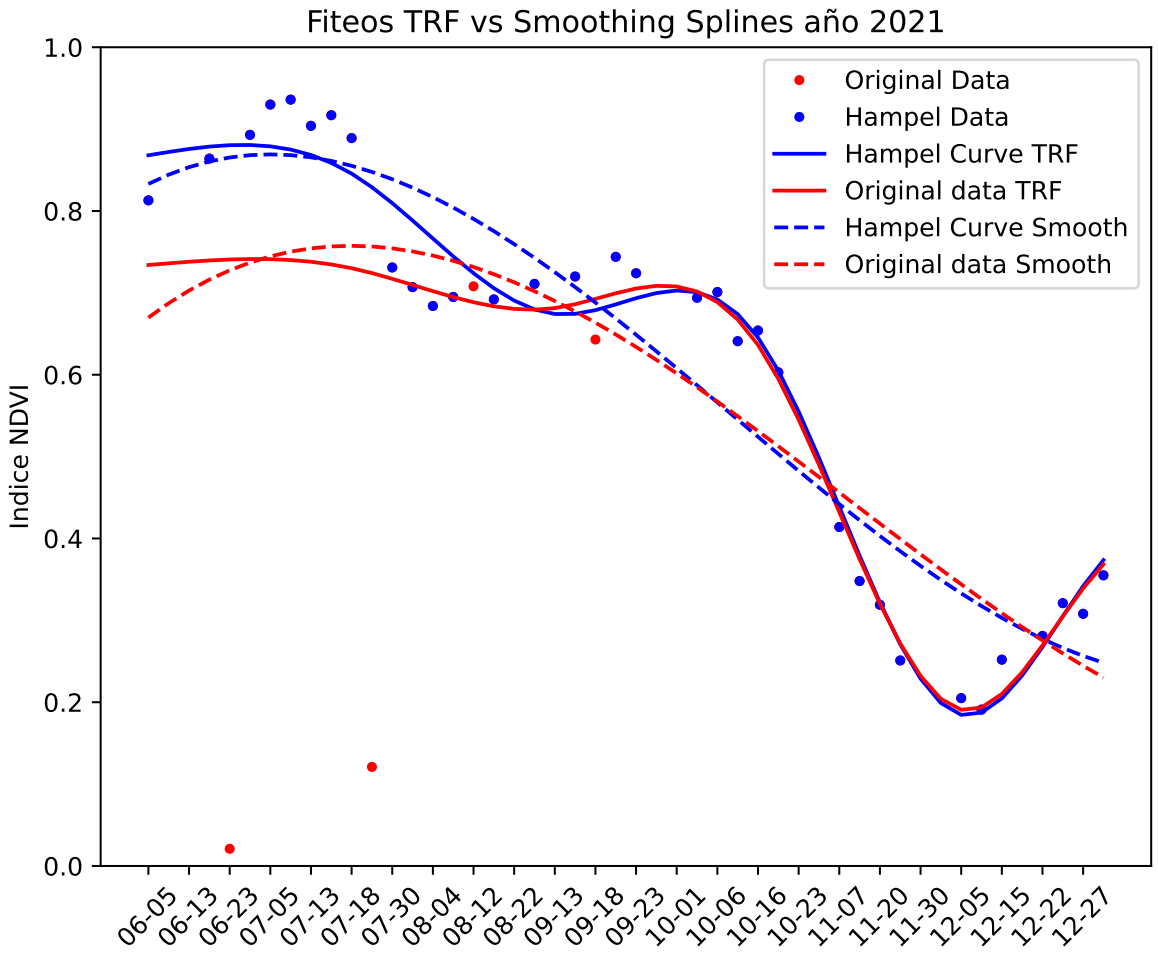

Fig 4.4: Fiteos Smooth Splines y TRF para datos de la serie de tiempo del índice NDVI del partido Adolfo Alsina provincia de Buenos Aires y datos filtrados con el método Hampel.

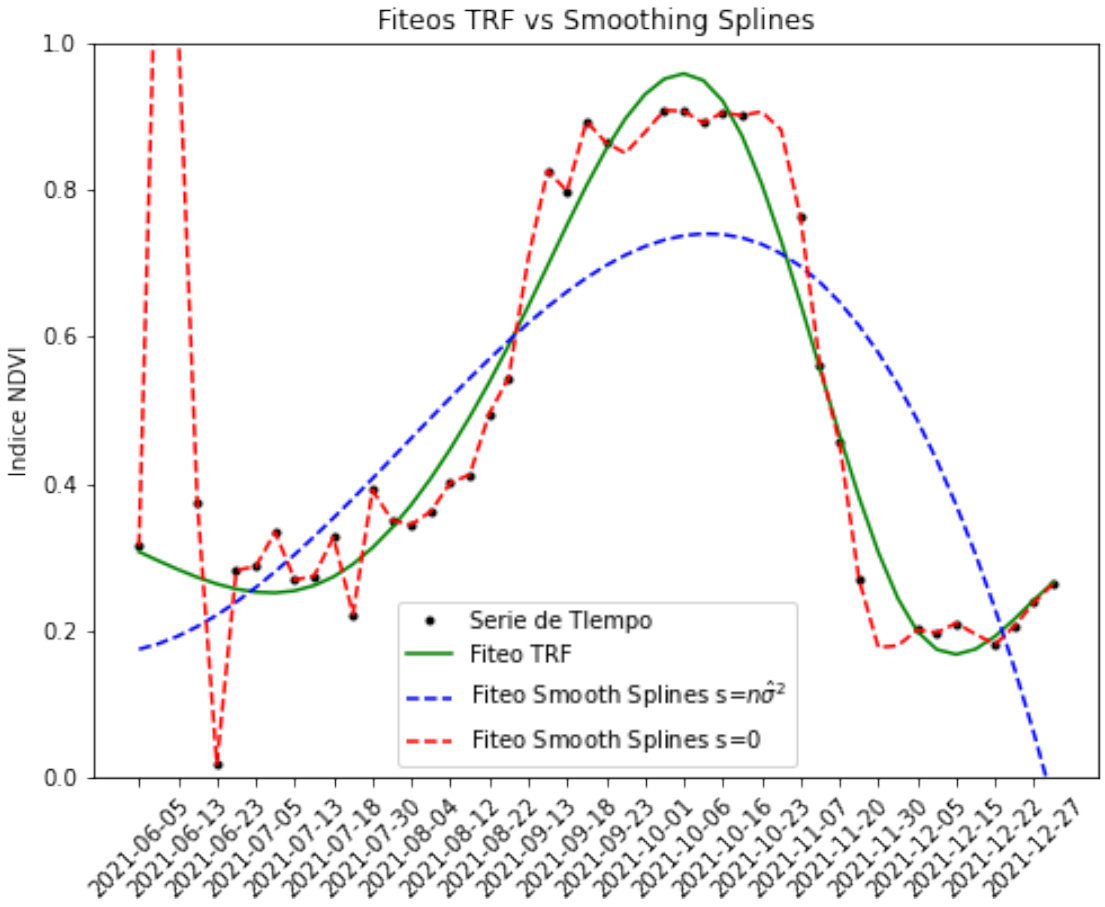

Fig 4.5: Fiteos de Smooth Splines y Splines-TRF para una serie de tiempo del indice NDVI del departamento Adolfo Alsina de la Provincia de Buenos Aires. Se presencia una serie de tiempo representativa del trigo.

El gráfico de la serie de tiempo con los fiteos estudiados figura 4.5 parece ser del cultivo del trigo pues su pico se manifiesta en los meses de invierno. Esta serie de tiempo fue tomada del departamento de Adolfo Alsina de la provincia de Buenos Aires en el año 2021.

4.3 Resultados con Clustering Robusto y No Robusto

Llevaremos a cabo TCLUST con parámetros fijos de $K = 6$ y $\alpha = 0.10$, el parámetro c lo tomamos igual a 10 las matrices de covarianzas se asumen por supuesto también distintas, por supesto que consideramos que cada cluster no tiene porque tener la misma cantidad de observaciones, además la cantidad máxima de iteraciones la tomamos igual a 150. Como se explicó en sección 3.3.2 el parámetro α representa la proporción de observaciones que podaremos y son consideradas parte del ruido. La cantidad de clusters está fijada por conocimientos de campo como se mencionó anteriormente sección 4.1. El parámetro c determinaba el cociente entre el autovalor más grande y el autovalor más pequeño de las matrices de covarianzas, con lo que mide que tan diferentes pueden ser las formas de los clusters. Salvo la cantidad de agrupaciones los parámetros de la función *tclust* fueron tomados manualmente. Se lleva a cabo el clustering para departamentos con más de 100 filas pues para menos observaciones no consideramos que se tenga una región representativa del espacio geográfico, existen departamentos a los cuales se les aplicó todo el proceso de limpieza (Filtro Hampel, eliminación manual de filas con más de un 80%) y despues extracción de parámetros y terminaron con menos de 100 observaciones, esto puede ser explicado porque son departamentos cuya superficie territorial no es muy considerable (y por lo tanto son bases de datos livianas con pocas filas) o porque los datos de índice NDVI en esos sitios son demasiado problemáticos (por errores de medición del satélite, nubes que bajan el índice NDVI o cualquier otro problema que no podamos controlar) o por las dos causas a la vez. La idea es ejecutar el aprendizaje en zonas donde existan datos *confiables* y no se gaste innecesariamente tiempo de corrida estimando resultados que no puedan ser del todo fiables.

Se presentarán las proporciones estimadas y las reales para el año 2021 en todos los departamentos analizados de este año. Analizando la figura 4.6 se ve una marcada mejora en las predicciones de la proporción cultivada de trigo usando el clustering robusto, es importante notar que en algunos partidos el modelo no robusto incluso ni siquiera logró identificar la presencia del Trigo para departamentos en los que evidentemente la proporción de este cultivo es considerable (con proporciones mayores al 15%).

Como datos de verdad de campo usamos los datos encontrados en:

https://datosestimaciones.magyp.gob.ar/reportes.php?reporte=Estimaciones

Para la lectura y carga de los datos de verdad de campo, así como en el cálculo de las superficies territoriales por departamento y el área sembrada de Trigo se recurre a la libreria de Python GeoPandas que se usa ampliamente para el manejo de datos geográficos. El archivo descargado es de tipo *shapefile*. Esta base de datos contiene como columnas el año, la Provincia, el departamento y el cultivo. Para hacer el estudio y encontar los datos concisos fue necesario filtrar los datos para obtener sintetizada la información y poder comparar las proporciones estimadas con las de campo.

El clustering No robusto (GMM) se lleva a cabo usando la librería de Scikit-Learn de Python, mientras que el Clustering robusto se lleva a cabo usando el software estadístico de acceso

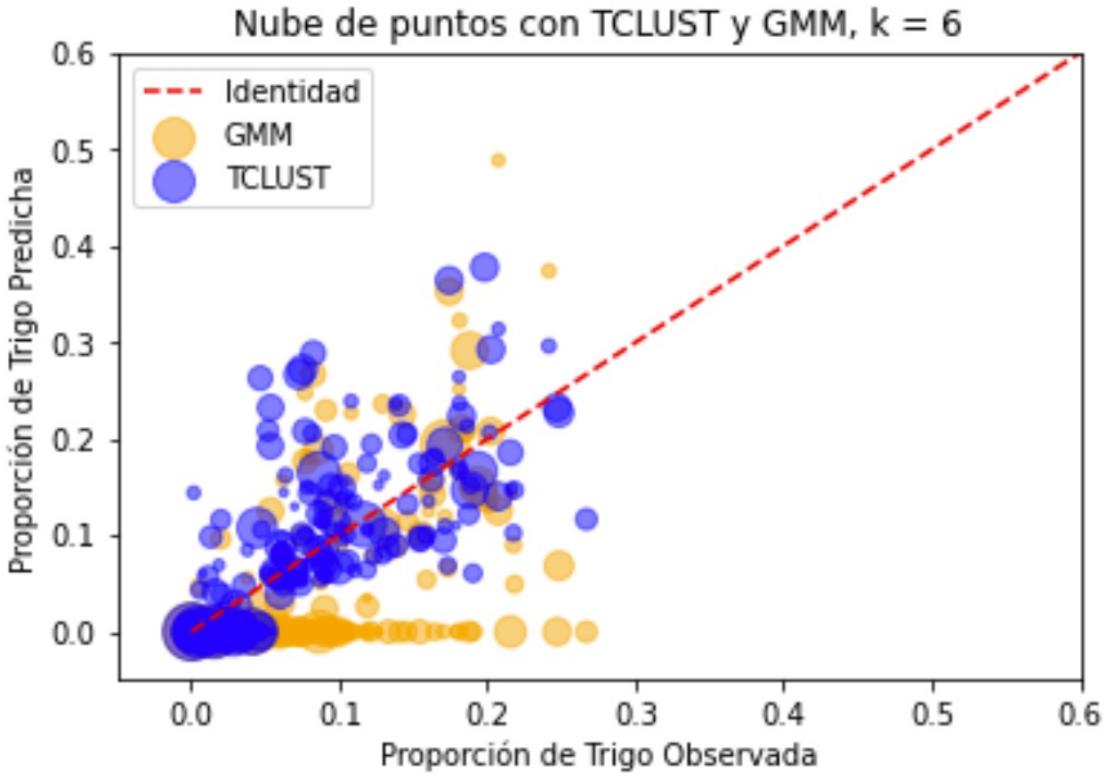

Fig 4.6: Proporciones predichas y observadas del Trigo usando GMM y TCLUST

libre R-Studio, más precisamente haciendo uso del paquete *tclust*, no encontramos una implementación en Python del algoritmo TCLUST. Como dato en tiempo de ejecución cuando aplicamos el algoritmo TCLUST con los parámetros especificados al departamento San Justo de la provincia de Córdoba (cuya superficie es de $15559 Km^2$, más grande que un país como Montenegro cuya superficie es de un poco menos de $14000 Km^2$) el tiempo de ejecución fue de 107 minutos, la precisión con TCLUST fue de: 0.00525 lo que nos da un resultado por lo menos para este departamento bastante preciso, a modo de prueba y como interés particular se intentaron llevar a cabo los algoritmos K-TAU centers y RMBC, basados en el concepto de escalas Tau introducido por Victor Yohai y Ruben H. Zamar (Yohai and Zamar 1988) e implementados en (González 2019) para este mismo departamento y se presentaron inconvenientes en la memoria, las implementaciones actuales no están optimizadas en cuanto al uso de memoria y por lo tanto no es aplicable en su forma actual a nuestro dataset. El algoritmo K-TAU centers está implementado en R-Studio se usa la función *improvedktaucenters* de la librería *ktaucenters*. Así mismo el algoritmo RMBC está implementado también en R-studio se usa la función *RMBC* de la librería *RMBC*.

No se detecta una clara presencia del Trigo usando GMM, pues no se observan picos mayores a 0.6 del NDVI figura 4.7 en el departamento General San Martín de la Provincia de Córdoba el cual por datos de campo tiene una proporción de un poco mas del 20% en el año 2021, concluyendo que GMM no logró captar el trigo para este departamento, por el contrario, si se detecta una clara presencia del Trigo figura 4.8 (cluster 2, color azul) usando el clustering robusto, también se manifiestan dos clusters (el 1 y el 6, colores rojo y magenta respectivamente) que parecen ser cultivos de verano pues parece que alcanzan picos de NDVI en los

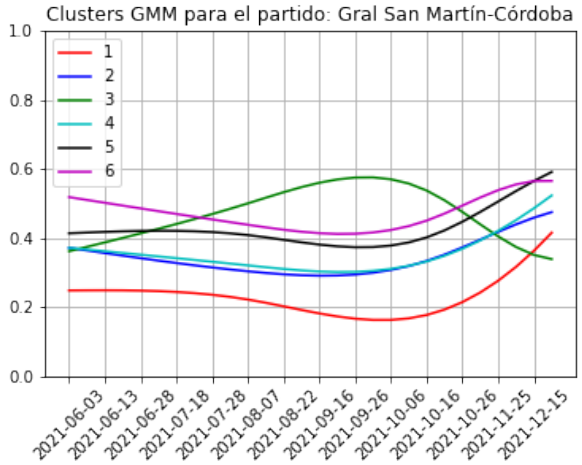

Fig 4.7: Centroides generados por GMM.

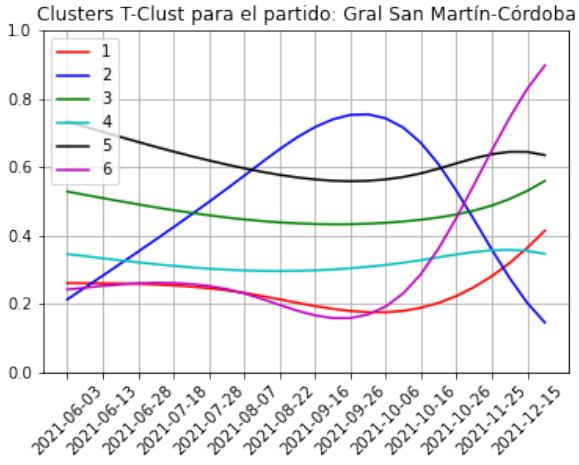

Fig 4.8: Centroides generados por TCLUST.

meses de diciembre, enero y febrero todo esto en el mismo departamento General San Martín de la Provincia de Córdoba para el año 2021, como vemos TCLUST ha logrado reconocer patrones más precisos, la proporción estimada es del 18.57% que si bien difiere al porcentage real en un 3% aproximadamente representa una evidente mejora en la fiabilidad de los resultados para la predicción de proporciones cultivadas para años futuros sin un respaldo de los datos de campo. El ruido presente en el modelo sesga claramente el resultado final y se debe recurrir necesariamente a múltiples tratamientos de robustez antes y durante el aprendizaje de clusterización. Calculamos el RMSE ponderado por superficie:

$$RMSE = \sqrt{\sum_{i=1}^{220} w_i(\hat{P}_i - P_i)^2},$$

donde, $\hat{P}_i$ = Proporción estimada y P_i = Proporción Real. Los pesos w_i están dados por:

$$w_i = \frac{\text{Superficie del i-ésimo departamento}}{\text{Superficie total}}.$$

Para un total de 220 departamentos analizados se obtiene usando clustering No robusto un error cuadrático medio poderado por superficie de 0.072 rms y para clustering robusto el error es de 0.059 rms mostrando una sustancial mejoría en el aprendizaje.

4.4 Posibles mejoras y próximos pasos

Para los departamentos con la mayor cantidad de error cuadrático medio se puede hacer un estudio particular y observar que pudo haber ocasionado esta predicción sesgada, proponer alternativas para estos partidos patológicos y observar si existen similitudes tanto geográficas, climáticas o de otra índole entre estas regiones que puedan generar esta diferencia elevada entre la proporción estimada de trigo en esta tesis y la proporción real de trigo cultivada. Se puede llevar a cabo este estudio tomando una región de superficie mas acotada y hacer foco

en una provincia (por ejemplo) hay que enfatizar que se analizan un aproximado total de 1.200.000 kilómetros cuadrados de superficie.

Una alternativa posible para la cual no habríamos eliminado observaciones atípicas del indice NDVI en las series de tiempo, podría haber sido detectarlos usando el filtro Hampel y posteriormente haber usado Smoothing Splines dándole a estas observaciones un peso muy bajo así de esta forma el fiteo no lo tendrá muy en cuenta, usando de esta forma Smoothing Splines que es una alternativa temporalmente más eficiente y no desechando valores del NDVI.

Las predicciones cultivadas realizadas para el trigo (cultivo de invierno) se pueden replicar para otros cultivos como lo pueden ser la soja (cultivo de verano) su relevancia radica dado que Argentina es el tercer productor mundial para datos tomados en el año 2022 (Lenz et al. 2022) con un estimado total de 43.86 millones de toneladas al año por detrás de Brasil y Estados Unidos, primero y segundo lugar respectivamente, también es importante remarcar que el complejo sojero argentino representó en 2021 el 30.6% de las exportaciones nacionales, por un monto de 23.841 millones de dólares (Schmidt 2022) y (Colussi et al. 2023). Así mismo se puede replicar el mismo estudio para un cultivo central como el maíz y otros más como la cebada.

5 Apendice A (Método de minimización de región de confianza)

El método de minimización con restricciones TRF (Trust region reflective) es un método iterativo de optimización numérica empleado para resolver problemas de programación no lineal, ampliamente usado en modelos de Machine Learning, especialmente para modelos de Deep Learning (Li et al. 2022) y (Hagn and Grau 2022). El objetivo en éste apéndice no es estudiar el método en profundidad, pero si presentar en general las ideas principales en las que se basa este método, y este enfoque de minimización basado en regiones de confianza. Se busca solucionar el siguiente problema:

$$\min_{x \in \mathbb{R}^n} f(x) : a \leq f(x) \leq b$$

Donde f debe ser una función suave (admite derivadas de cualquier orden en $\mathbb{R}$), con codominio igual a $\mathbb{R}$. Es decir, $f : \mathbb{R}^n \to \mathbb{R}$.

El método en cada iteración define un vecindario alrededor de la solución actual, es decir creamos una *región de confianza* en cada paso, para la cual calculamos una aproximación de la función original, en general de orden cuadrático, a la que le encontramos el óptimo local sobre esa región. (Le et al. 2016)

La región de confianza la definimos como: $L = \{x \in \mathbb{R}^n : \|x - c\|_2 \leq \delta\}$

Los métodos de región de confianza de alguna manera se pueden asemejar con los métodos de búsqueda lineal, los cuales escogen una dirección de paso y el tamaño del paso, por el contrario en métodos de región de confianza escogemos una región y un punto de mejora dentro de esa región previamente tomada. Para encontrar la solución óptima, los principales pasos a seguir son los siguientes.

Definición de región de confianza: Se define una región de confianza alrededor de la iteración actual donde se espera que el modelo cuadrático sea preciso. En general, el tamaño de la región de confianza depende del radio de la zona de confianza, que es modificado adecuadamente durante el proceso de optimización. La región de confianza actúa como un tamaño de paso y garantiza que el algoritmo no se desvíe demasiado de la iteración actual.

Formulación del subproblema: En la región de confianza, se formula un subproblema para encontrar la dirección del paso que minimiza el modelo cuadrático bajo la restricción del dominio de confianza. Este subproblema se puede formular como un problema con optimización mejorada o un problema sin restricciones con penalización. Se pueden utilizar diferentes métodos para resolver el subproblema, como el método del gradiente conjugado o El método de Newton, existen esfuerzos en la literatura para resolver este problema (Yuan 2000).

Actualización de dirección de paso: Una vez que se obtiene la dirección del paso de la solución del subproblema, esta se utiliza para actualizar la iteración actual. El tamaño del paso se determina minimizando la función objetivo en el dominio de confianza y el acuerdo entre el modelo y los valores funcionales reales.

Proceso iterativo: Se repite la construcción de un modelo cuadrático, la definición de una región de confianza, la formulación de un subproblema. se iteran hasta que se cumplan los criterios de parada. Los criterios de parada pueden basarse en lograr un cierto nivel de tolerancia o número de iteraciones.

Los pasos se pueden escribir de la siguiente manera:

1. Inicializamos el umbral de tamaño: δ, la solución inicial x_0^*, y $n = 0$, n se refiere al número de iteración del algoritmo.
2. Resolvemos $x_n^* = \arg\min_x \mathrm{m(x)}$ sujeto a: $\|x - x_{n-1}^*\| \leq \delta$

Donde: $m(x) \simeq f(x)$, m esta dado por el polinomio de Taylor de segundo orden:

$$\mathrm{m(x)} = f(c) + \nabla f(c)(x - c) + \frac{1}{2}(x - c)^T H(c)(x - c)$$

$\nabla f(c)$ es el vector gradiente y H es la matriz Hessiana.

3. Actualización de la región de confianza: Ajustamos el radio de la zona de confianza según el *éxito* del paso. Más precisamente calculamos la relación ρ entre reducciones reales y reducciones previstas de la siguiente manera:

$$\rho = \frac{f(x_k) - f(x_k + \Delta x)}{\mathrm{m}(x_k) - \mathrm{m}(x_k + \Delta x)}$$

Donde: $f(x_k) - f(x_k + \Delta x)$ es la reducción real de la función objetivo.

Donde: $\mathrm{m}(x_k) - \mathrm{m}(x_k + \Delta x)$ es la reducción aproximada en la función objetivo según el modelo cuadrático

La región de confianza puede aumentar cuando la relación ρ es cercana a 1, es decir se acepta el paso, por lo tanto iteramos y actualizamos. Si ρ es mucho menor que 1 entonces se omite el paso y el área de confianza debe reducirse para centrarse en el área local.

4. Verificamos la convergencia en función de criterios de terminación, como alcanzar cierto nivel de tolerancia o cierta cantidad de iteraciones.

Durante los años, el enfoque para aproximar la zona de confianza ha sido estudiado y modificado derivando en distintos variaciones, las más conocidas son el método Dogleg y El método de región reflectiva de confianza, para este trabajo usaremos la segunda opción, pues resulta ser la más apropiada para problemas con restricciones, el algoritmo actualiza la iteración para satisfacer la restricción aceptando primero un paso en el dominio de confianza y luego proyectándolo en el conjunto factible, en caso de ser necesario.

6 Apendice B (Quarto)

Quarto es un sistema de publicación científica y técnica de código abierto en el cual se puede crear contenido utilizando cuadernos Jupyter o con Markdown de texto sin formato en cualquier editor, Esta tesis fue escrita en el entorno Quarto y particularmente usamos *Visual Studio Code* como editor de texto, Quarto se presenta en gran medida como la continuación del entorno R-Markdown también muy ampliamente usado en R-Studio, sus similitudes son evidentes.

Se puede crear contenido dinámico con lenguajes de programación y sofwares estadísticos usados en la ciencia de datos y en la investigación científica como lo son: Python, R y Julia. Se permite en este entorno compilar código $LaTex$ para la redacción de textos matemáticos y así mismo usar codigo Python, R o Julia combinados, lo que lo hace de particular interés y utilidad.

Quarto nos permitió generar simulaciones en código R y en código Python dentro del mismo entorno permitiendo usar funcionalidades compartidas, como se mencionó anteriormente en los capítulos 3 y 4 la libreria que lleva a cabo el clustering robusto TCLUST no está implementada en Python, por lo que se debió recurrir al uso de los dos lenguajes (R y Python) necesariamente. Para la formalización matemática se uso el leguaje $LaTex$.

El tratamiento de la abundante cantidad de datos con el filtro Hampel, los fiteos llevados a cabo como lo son Smooth Splines y TRF así como el aprendizaje No supervisado robusto y no sobusto se llevaron a cabo en archivos .py y ,ipynb en el entorno de Jupyter notebook y con el uso de instancias en la nube de Amazon AWS, Las imágenes y resultados fundamentales de la tesis como lo son la nube de Puntos de las proporciones estimadas por los dos modelos se almacen en Quarto y se presentan en la tesis, para simulaciones básicas y la escritura de este documento se uso el propio entorno de Quarto.

Bibliografía

Akinshin, Andrey. 2022. "Finite-Sample Bias-Correction Factors for the Median Absolute Deviation Based on the Harrell-Davis Quantile Estimator and Its Trimmed Modification." *arXiv Preprint arXiv:2207.12005*.

Banfield, Jeffrey D, and Adrian E Raftery. 1993. "Model-Based Gaussian and Non-Gaussian Clustering." *Biometrics*, 803–21.

Bickel, Peter J. 1965. "On Some Robust Estimates of Location." *The Annals of Mathematical Statistics*, 847–58.

Bishop, Christopher M. 2006. "Pattern Recognition and Machine Learning." *Springer Google Schola* 2: 645–78.

Bollen, Kenneth A. 2002. "Latent Variables in Psychology and the Social Sciences." *Annual Review of Psychology* 53 (1): 605–34.

Colussi, Joana, Nick Paulson, Jim Baltz, Carl Zulauf, Silvina María Cabrini, Maria Cecilia Paolilli, and Francisco Antonio Fillat. 2023. "Brasil Bate récord de Exportación de Soja y Desplaza a Argentina En El Mercado Mundial de Harina de Soja." EEA Pergamino, INTA.

Coretto, Pietro, and Christian Hennig. 2016. "Robust Improper Maximum Likelihood: Tuning, Computation, and a Comparison with Other Methods for Robust Gaussian Clustering." *Journal of the American Statistical Association* 111 (516): 1648–59.

Daffertshofer, Andreas, Claudine JC Lamoth, Onno G Meijer, and Peter J Beek. 2004. "PCA in Studying Coordination and Variability: A Tutorial." *Clinical Biomechanics* 19 (4): 415–28.

Danilov, Mike, Víctor J Yohai, and Ruben H Zamar. 2012. "Robust Estimation of Multivariate Location and Scatter in the Presence of Missing Data." *Journal of the American Statistical Association* 107 (499): 1178–86.

Dierckx, Paul. 1975. "An Algorithm for Smoothing, Differentiation and Integration of Experimental Data Using Spline Functions." *Journal of Computational and Applied Mathematics* 1 (3): 165–84.

Donders, A Rogier T, Geert JMG Van Der Heijden, Theo Stijnen, and Karel GM Moons. 2006. "A Gentle Introduction to Imputation of Missing Values." *Journal of Clinical Epidemiology* 59 (10): 1087–91.

Estivill-Castro, Vladimir, and Jianhua Yang. 2000. "Fast and Robust General Purpose Clustering Algorithms," 208–18.

Farcomeni, Alessio, and Antonio Punzo. 2020. "Robust Model-Based Clustering with Mild and Gross Outliers." *Test* 29 (4): 989–1007.

Fernández, Santiago Fernández, José María Cordero Sánchez, Alejandro Córdoba, and Alejandro Córdoba Largo. 2002. *Estadística Descriptiva*. Esic Editorial.

Fritz, Heinrich, Luis A Garcia-Escudero, and Agustin Mayo-Iscar. 2012. "Tclust: An r Package for a Trimming Approach to Cluster Analysis." *Journal of Statistical Software* 47: 1–26.

Gallegos, María Teresa, and Gunter Ritter. 2005. "A Robust Method for Cluster Analysis."

Garcia-Escudero, Luis Angel, and Alfonso Gordaliza. 1999. "Robustness Properties of k

Means and Trimmed k Means." *Journal of the American Statistical Association* 94 (447): 956–69.

García-Escudero, Luis A, Alfonso Gordaliza, Carlos Matrán, and Agustin Mayo-Iscar. 2008. "A General Trimming Approach to Robust Cluster Analysis."

García-Escudero, Luis Angel, Alfonso Gordaliza, Carlos Matrán, and Agustín Mayo-Iscar. 2010. "A Review of Robust Clustering Methods." *Advances in Data Analysis and Classification* 4: 89–109.

Godichon-Baggioni, Antoine, and Stéphane Robin. 2024. "A Robust Model-Based Clustering Based on the Geometric Median and the Median Covariation Matrix." *Statistics and Computing* 34 (1): 55.

González, Juan Domingo. 2019. "Métodos de Clustering Robustos." PhD thesis, Universidad de Buenos Aires. Facultad de Ciencias Exactas y Naturales.

Gutmann, Michael, and Aapo Hyvärinen. 2010. "Noise-Contrastive Estimation: A New Estimation Principle for Unnormalized Statistical Models." In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 297–304. JMLR Workshop; Conference Proceedings.

Hagn, Korbinian, and Oliver Grau. 2022. "Optimized Data Synthesis for DNN Training and Validation by Sensor Artifact Simulation." In *Deep Neural Networks and Data for Automated Driving: Robustness, Uncertainty Quantification, and Insights Towards Safety*, 127–47. Springer International Publishing Cham.

Khosravi, Pasha, Antonio Vergari, YooJung Choi, Yitao Liang, and Guy Van den Broeck. 2020. "Handling Missing Data in Decision Trees: A Probabilistic Approach." *arXiv Preprint arXiv:2006.16341*.

KRIEGLER, Frank J. 1969. "Preprocessing Transformations and Their Effects on Multspectral Recognition." In *Proceedings of the Sixth International Symposium on Remote Sesning of Environment*, 97–131.

Le, Thu, Behzad Fatahi, Hadi Khabbaz, and Wenjing Sun. 2016. "Numerical Optimization Applying Trust-Region Reflective Least Squares Algorithm with Constraints to Optimize the Non-Linear Creep Parameters of Soft Soil." *Applied Mathematical Modelling* 16 (October): 1–21. https://doi.org/10.1016/j.apm.2016.08.034.

Lenz, Desiree, Juan Sebastián Pappalardo, Adriana Alejandra Pazos, Federico Esteban Mutti, Guillermo Santiago Gallardo, Alejandro Gabriel Aparicio, Ana Laura Pietrantuono, Carlos Alejandro Rauque Perez, Mariana Murano, and Valeria Cristina Fernandez Arhex. 2022. "Piensos a Base de Insectos Para Alimentación Animal."

Li, Na, Donald M Arnold, Douglas G Down, Rebecca Barty, John Blake, Fei Chiang, Tom Courtney, Marianne Waito, Rick Trifunov, and Nancy M Heddle. 2022. "From Demand Forecasting to Inventory Ordering Decisions for Red Blood Cells Through Integrating Machine Learning, Statistical Modeling, and Inventory Optimization." *Transfusion* 62 (1): 87–99.

Lugosi, Gabor, and Shahar Mendelson. 2021. "Robust Multivariate Mean Estimation: The Optimality of Trimmed Mean."

Masud, Md Abdul, Joshua Zhexue Huang, Chenghao Wei, Jikui Wang, Imran Khan, and Ming Zhong. 2018. "I-Nice: A New Approach for Identifying the Number of Clusters and Initial Cluster Centres." *Information Sciences* 466: 129–51.

Molfese, ER, and N Fritz. 2020. "Production and Quality of Durum Wheat (Triticum Turgidum l. Subsp. Durum) in Argentina: Analysis of the 2014/2018 Five-Year Period."

Peel, David, and Geoffrey J McLachlan. 2000. "Robust Mixture Modelling Using the t Distribution." *Statistics and Computing* 10: 339–48.

Rey, William JJ. 2012. *Introduction to Robust and Quasi-Robust Statistical Methods*. Springer Science & Business Media.

Rodríguez, Corina Iris, María Macarena Arrien, Santiago Hernán Silva, and Maite M Aldaya. 2024. "Global Relevance of Argentinean Rainfed Crops in a Climatic Variability Context: A Water Footprint Assessment in Buenos Aires Province." *Science of The Total Environment* 927: 171946.

Ruppert, David, and Raymond J Carroll. 1980. "Trimmed Least Squares Estimation in the Linear Model." *Journal of the American Statistical Association* 75 (372): 828–38.

Salvia, M Mercedes, Nilda Sánchez, María Piles, Romina C Ruscica, Ángel González-Zamora, Esteban Roitberg, and José Martínez-Fernández. 2021. "The Added-Value of Remotely-Sensed Soil Moisture Data for Agricultural Drought Detection in Argentina." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 14: 6487–6500.

Savitzky, Abraham., and M. J. E. Golay. 1964. "Smoothing and Differentiation of Data by Simplified Least Squares Procedures." *Analytical Chemistry* 36 (8): 1627–39. https://doi.org/10.1021/ac60214a047.

Schmidt, Juliana Victoria. 2022. "Análisis de Los Efectos de Los Derechos de Exportación Sobre La Soja y El Maíz En Las Decisiones de Los Productores."

Schubert, Erich, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. 2017. "DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN." *ACM Transactions on Database Systems (TODS)* 42 (3): 1–21.

Sued, Mariela, Marina Valdora, and Víctor Yohai. 2020. "Robust Doubly Protected Estimators for Quantiles with Missing Data." *TEST* 29 (3): 819–43.

Sued, Mariela, and Victor J Yohai. 2013. "Robust Location Estimation with Missing Data." *Canadian Journal of Statistics* 41 (1): 111–32.

Yohai, Victor J, and Ruben H Zamar. 1988. "High Breakdown-Point Estimates of Regression by Means of the Minimization of an Efficient Scale." *Journal of the American Statistical Association* 83 (402): 406–13.

Yuan, Ya-xiang. 2000. "A Review of Trust Region Algorithms for Optimization." In *Iciam*, 99:271–82. 1.

Zhao, Ruofei, Yuanzhi Li, and Yuekai Sun. 2020. "Statistical Convergence of the EM Algorithm on Gaussian Mixture Models."